

hard core

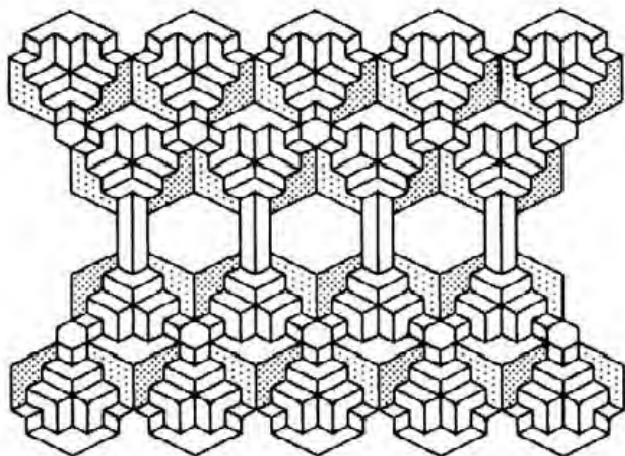
THE JOURNAL
OF THE
BRITISH APPLE
SYSTEMS
USER GROUP



October 1984

£1

VOLUME 4 No. 5



HOW TO FIND Lux Computer Services Ltd

COLONIAL WAY, WATFORD, HERTS. WD2 4AT

TEL WATFORD (0923) 44617

Call in now for a Macintosh demonstration

mickie

A special purpose language for multiple choice tests and quizzes, questionnaires and Computer aided learning (CAL) - £50 + VAT

★ MICKIE programs can be written by people with no previous experience of programming. Many people have neither the inclination nor the aptitude to master general purpose languages such as BASIC or Pascal.

★ MICKIE can be used by people who have never used or even seen a computer before. This is demonstrated by MICKIE's success with hospital patients.

★ MICKIE is written in a simple to use, easy to remember format, designed specifically for the first time user who does not want to know more than is absolutely necessary about computers and computer languages.

★ Originally developed for medical history taking MICKIE has been evaluated in hospitals, schools and commerce. It saves time and provides comprehensive, legible and structured records.

★ The original MICKIE was developed at the National Physical Laboratory by the late Dr Christopher Evans (well-known as the author of 'The Mighty Micro' and 'The Making of the Micro').

Full Apple implementation from

SYSTEMICS LIMITED

21-23, The Bridge, Harrow, Middlesex HA3 5AG.

Tel: 01-863 0079



MICKIE is a trademark of the National Physical Laboratory



BASF DISKS

- individually numbered
- 5.25" single sided, single density
- suitable Apple][, //e & ///
- box contains 10 disks

1 BOX £ 15.50 NO

10 BOXES £140.00 EXTRAS

WE PAY VAT, POSTAGE & PACKING

Send order & remittance to:

SYSTEMICS LIMITED,

21-23 THE BRIDGE

HARROW

MIDDLESEX HA3 5AG

01 863 0079

Access & Barclaycard accepted

hardcore

THE JOURNAL OF

THE BRITISH APPLE SYSTEMS USER GROUP

P.O. BOX 174 WATFORD WD2 6NF

BASUG Ltd. is a non-profit-making company limited by guarantee.

Basug Committee

Bob Raikes	Chairman
Norah Arnold	Secretary
Roger Gear-Evans	Treasurer
Graham Attwood	Software Library
Richard Beck	Courses
Richard Boyd	Meetings, Local Groups
Keith Chamberlain	Membership
Tony Game	Bulletin Board
Roger Harris	Literature Library
Jim Panks	Software Library
Quentin Reidford	
John Rogers	Special Release Software
Peter Trinder	Updates
Ewen Wannop	Prestel
Fran Teo	Administrator
Yvette Raikes	Hardcore Editor
Daphne Jenkins	
Peter Blair &	
Jim Panks	Assistants

BASUG Telephone No.: Newbury (0494) 444444.
Please note: This number is administration ONLY.

Bulletin Board: (0494) 444444.

Founder members of the Association of Computer Clubs.

CONTENTS

4 Editorial	
Lonely Apples	
Chairman's Corner	
5 FORTH	
9 User Hostility	Hugh Dobbs
10 Assembler Review	Bob Raikes
13 Mac Notes	Seth Proctor
14 Applewriter //e	Peter Knight
15 Epson DX100	Dick Menhinick
16 BASUG Telephone Numbers	Jim Panks
18 Quiz Writing	Hugh Dobbs
20 String Print	R. C. Lowe
Software Library	
22 Beginners' Page	John Sharp
23 Courses	
25 Local Groups	
26 Book Reviews	
30 Keyboard Lowercase Mod.	Dick Menhinick
32 Children's Software	
G. G. Wood & Elizabeth Raikes	
33 Administration	
34 BASIC Note	Adam Broun
Education	Norah Arnold
37 Lister	Roger Harris
38 DS:3	Bob Raikes
42 Sorting	R. C. Lowe
Readers' Letters	
45 Reinventing the Wheel	Yvette Raikes
47 Diary	



**INTERNATIONAL
APPLE CORE**
TM

MEMBER OF THE
INTERNATIONAL APPLE CORE

HARDCORE is produced using Applewriter II, and printed on an APTEC Flowriter with a Madeleine Daisy-wheel

Front Cover: An optical illusion produced on the Macintosh by Norah Arnold.

COPYRIGHT (C) - The contents of this journal are copyright of the British Apple Systems User Group and /or the respective authors. However permission will normally be granted for non-commercial reproduction by user groups affiliated to the International Apple Core, provided the author and source are properly credited.

The opinions and views expressed are those of the various contributors, and are not necessarily supported by the British Apple Systems User Group.

Editorial

Thank you to those who responded to the request for help with utilities. We have Roger Harris's programming aid "Lister" in this issue. If any more of you want to tell us about your favourite utility - either one you have written yourself or a commercial one - please write in. You don't need to write a lot, just what it does and why you like it.

I also have some books for review, two on Visicalc and one each on the Apple /// and CP/M. Any takers?

Unfortunately, the Expert Systems article has had to be held over again but it will appear in the next issue along with a review of a commercial Expert System.

Do remember that the copy date for the Christmas edition is earlier than usual. Please get your items in by October 26th. We expect to put out a bumper edition then but this does not mean that there will be space waiting for late entries. Please will all those who currently have books or other items for review ensure that these are reviewed for the next issue or that, where this is not possible, you have let me know. Most of the people who donate their products for review want to see them reviewed for Christmas. Try not to let them down.

WANTED

Have any of you longed for a chance to be the first to read what goes in Hardcore? Well, here is your chance. We are looking for a new editor of Hardcore to take over from the end of the year. The job is not difficult but does take up quite a lot of time particularly in the fortnight following copy dates. Anyone willing to take on the job will be given the opportunity to see the Christmas edition produced and will be given lots of help and support during the production of their first issue if they need it. Anyone who wishes to apply or to ask for further details should telephone Bob Raikes on 0252 370621.

Lonely Apple

The following member would like to get in touch with people who use Apple II+ or U-computers in Norway:

Øyvind Øyvind
Trondheim 26
000000 000000

Chairman's Corner

Our new committee members are now finding their feet, and progress is being made. Volunteers in the Software group are currently working on moving the Software Library from DOS 3.2 to 3.3. The machine Code course in October will happen, and a Logo course is promised. Mid-Apple are hosting a meeting with help from Richard Boyd on Communications, and there are 5 new releases in the Special Release Software series in the course of preparation.

There have been 2 moves of note. Fran Teo has moved recently, apologies if there has been any delay in meeting orders etc. Those who know Fran will realise how unsettled her circumstances have been. We wish her the best of luck in her new home.

The Bulletin board has also moved. Its founding father Quentin Reidford has taken the plunge in trying to go commercial. There is a board still on his number, but it is no longer the BASUG one. Tony Game has taken over the board, and Peter Trinder has stepped into his shoes as Press Officer. If you have any news that may be of interest, or an unusual application for your Apple, please let him know.

As you will see, Yvette is relinquishing her position as editor of Hardcore after the next issue. We are urgently looking for someone to step into the breach. Anyone interested should contact myself, Norah Arnold or Jim Panks. There is a great deal of work involved, but the job is rewarding in the personal and financial sense. Apply NOW.

At BASUG we aim to be both amateur and commercial. If our standards occasionally fall below those that both you expect and that we expect of ourselves, please let us know. Remember BASUG is run on YOUR behalf. If on occasion we seem not to be acting with this in mind, please let myself or someone else on the committee know.

WARNING.

There have been problems with the 64k Printer Buffer Card from Peanut. It wouldn't work with Visicalc or Logo. Peanut have offered to fix it. Anyone else had any problems?

Apple users need Hardcore to do it.

FORTH

A Quick Introduction to Forth on the APPLE

by Hugh Dobbs

Newtown School, Waterford, EIRE.

The computer package supplied to schools by the Department of Education included a variety of languages: four versions of BASIC, Comal, Pascal, 8080 Assembler, 6502 Assembler, PILOT. This paper is a brief argument in favour of one language which was not included: Forth.

I have used BASIC extensively, especially APPLESOFT; it is an easy language to write in, and mistakes take little time to correct if you can find them, but most versions lack any structure more advanced than a subroutine, and most are interpreted and therefore slow.

I have used Pascal; in it, program development is much slower since it is compiled and there is a lot of time spent switching between compiler and filer, reloading source code and editing it, and so on; Pascal programs can be highly structured and, being semi-compiled, run fairly fast. The UCSD version of Pascal also has the advantage of being portable over many different computer systems. The language is however fussy in the punctuation it requires and in the need for the declaration of data types and so on.

I have also used 6502 Assembler language extensively. It is very hard to get things right in it; it is unstructured and has none of the features of higher-level languages; but it is very fast indeed when running.

Somewhere in the middle, there is a need for a compromise, especially in applications where computers control machinery: you need the system to be able to make decisions quickly, but also you need to know what each part of your program does and to be able to change sections of it easily without affecting others. One suitable language is Forth, which is fast but structured, combines the ease of a high-level language with the speed of machine code (very nearly), is both an interpreter and a compiler, and is available on a wide range of systems. Since versions of it are in the public domain most Forth systems are fairly cheap (often under £50), and there is a fair degree of standardisation with only two

major dialects and a few eccentric versions which conform to neither.

The version I use on the APPLE is from the Forth interest group in USA, written by Kuntze but typed in and assembled by me, and it still has bugs in it but is on disk no. 84 in the software library if anyone wants to try it out. I am also using GraFORTH for its turtle-graphics, and evaluating Metacrafts FORTH which is a powerful and well-thought-out version with overlays. I also have Forth running on a BBC machine (r q Forth, Acornsoft and the JWB-FORTH ROM), on Atoms (Acornsoft), on a ZX-81 (Artic), on a Spectrum (HL-FORTH) and naturally on an Ace, and I am collecting information on other implementations.

The APPLE version starts up when its disk is booted, clearing the screen and announcing "THE 4TH APPLE!". What follows is a condensation of what happens in about three periods' worth of Computer Studies (whether at first year, second year or fifth year level).

Forth, like any other language, has a certain vocabulary. Unlike most other computer languages, but like natural languages, it grows by the addition of new words to its vocabulary. The new words are defined in terms of the existing words and thereafter can be used in exactly the same way as the older words - in fact a large part of the Forth nucleus is made up of words which have been defined in Forth rather than in machine code.

Most words are defined using the Forth word [:] (colon). A word in Forth can be made up of any sequence of characters which does not include a space, an ASCII NULL, or an ASCII CR character, and here the word is just a colon. By general agreement, when a Forth word is likely to cause confusion in a typed article (i.e. in an English sentence), it may be enclosed in [squarebrackets]. Here is an example of a 'colon definition'; lots more will follow:

```
: BELL 7 EMIT ;
```

Here the colon says 'I am going to teach you a new word using words you know already'. The new word is to be called [BELL], and the colon puts a header into the vocabulary with the word's name BELL, a link pointing to the most recently defined word before that, and an address which directs the interpreter to handle BELL as a word which has been defined

using [;]. It then turns on the compiler, which processes the rest of the line. The [7] is not found in the vocabulary, so it is construed as a number and stored as such. [EMIT] is a word in the Forth nucleus and is therefore compiled. The [;] says 'that is the end of the definition', and its main effects are to close the new definition, to mark the new word so that it can be found in a vocabulary search, and to turn off the compiler. The interpreter has then reached the end of your instruction, so it prints 'OK' and waits for you to do something else.

If you now type BELL and press 'return', the interpreter searches its vocabulary (starting with the most recently-defined word), finds [BELL], and executes it. This produces a number 7 which EMIT converts to ASCII code 7 and sends to the current output device; unless you are using a printer, this produces a 'bell' noise on the speaker.

If you type BELL BELL BELL and press 'return', the bell rings three times before the interpreter responds with 'OK'. So far the transactions look like this:

```
THE 4TH APPLE!
: BELL 7 EMIT ; OK
BELL OK
BELL BELL BELL OK
```

because the interpreter absorbs your 'return' without actually starting a new line.

Now that BELL has been defined it is in the Forth vocabulary and can be used in the definition of further words; for example, supposing you want to produce seven bells you could always type BELL seven times, but you can also do

```
: BELLS 0 DO BELL LOOP ;
```

after which

```
7 BELLS
```

will produce the desired result and give the added benefit of being able to have 100 BELLS or even 32767 BELLS if you wish. What happens here is that the DO...LOOP inside BELLS counts up from 0 to whatever number comes before it (7 or 100 or 32767), and whatever lies between DO and LOOP gets done so long as the counter has not reached the limit.

Computers are often used for simulations. Not having any special input/output hardware, I

go in for simulated simulations. For instance, suppose I want to simulate an alarm clock or a burglar alarm. When the alarm goes off, a bell should ring until someone does something about it. Something which can be simulated by pressing a key on the keyboard, for instance. So I type

```
: ALARM BEGIN BELL
```

- if I press 'return' at that point the interpreter does not respond with its usual 'OK' because I have not finished my definition yet (no [;] yet). After ringing the bell, it should test to see whether anyone has pressed a key

```
?TERMINAL
```

(or in some versions of Forth ?ESC, etc) - this checks and leaves a true-or-false flag. I want the BELL ?TERMINAL sequence to continue until the result is true, so

```
UNTIL ;
```

completes the definition. Now typing ALARM will set it off. The burglar alarm type would have to be a bit more resistant to being turned off, so I might use

```
: BURGLAR BEGIN BELL AGAIN ;
```

in which case BURGLAR will start an infinite loop which can only be stopped by a machine reset or by turning off the power (don't try that until the end of the session!).

To some extent this illustrates the structured nature of Forth; it is worth mentioning that any branching construct such as DO...LOOP or BEGIN...AGAIN or BEGIN...UNTIL or IF...ELSE...THEN can only be used within a colon definition (or equivalent), which means that it is impossible for branches to occur outside the current structure (compare the typical BASIC nastiness of GOTO from inside a subroutine or from inside a FOR...NEXT loop). Also the compiler checks the syntax to ensure that the branching structures themselves are properly constructed.

Now I want to illustrate something of the speed involved. BASIC is relatively slow, and probably most people have tried the traditional APPLE squawk program

```
10 X = PEEK (-16336) : GOTO 10
```

and know how low a note that produces; even the variant


```
10 S = -16336
20 X = PEEK (S) : GOTO 20
```

gives a note which is not high enough to be satisfyingly musical, so most game programs in BASIC include a machine-code routine for producing notes. Forth is fast enough to do it alone (if you don't mind whether the notes correspond to actual musical frequencies or not). A simple square wave output from the APPLE's speaker can be generated by producing a series of clicks at regular intervals, i.e. click pause click pause So I define

```
:CLICK -16336 C@ DROP ;
```

That defines [CLICK] to produce a number -16336, go and get the contents of that memory address (thus clicking the speaker) and DROP that value so as to end up with nothing (except a click).

```
:PAUSE 0 DO LOOP ;
```

That is like [BELLS] above, in that it counts up from 0 to whatever number you feed it (less 1); but inside the loop it does nothing. So

```
10000 PAUSE
```

says 'go into a corner and count up from 0 to 9999'; it takes much less time than BASIC's FOR A = 0 TO 9999 : NEXT A.

```
:NOTE 100 0 DO DUP PAUSE CLICK LOOP
DROP ;
```

That defines [NOTE] as a word which does something 100 times; what it does is take a DUPLICATE copy of whatever number it was fed, do that length of a PAUSE, then a CLICK, and loop back again so that it produces 50 complete wave cycles at some unknown frequency.

```
1 NOTE
```

produces the highest possible note of this type, and

```
1 NOTE 2 NOTE 3 NOTE
```

will give a descending pattern. We can build up a 'scale' by feeding a decreasing sequence of numbers to NOTE:

```
:SCALE 0 20 DO 1 NOTE -1 +LOOP ;
```

Here [1] is a word which gives the value of the loop counter (to feed to NOTE), and

since we want to count backwards I have to use [-1 +LOOP] which says 'count down 1 at a time', rather than simply LOOP. Now try SCALE then

```
: DOWNSCALE 21 1 DO 1 NOTE LOOP ;
```

```
SCALE DOWNSCALE SCALE
```

and so on. It is not very musical, but great fun to experiment with. Further words can be built up using these, such as

```
: PRACTICE BEGIN SCALE DOWNSCALE
?TERMINAL UNTIL ;
```

- the effect of that should by now be predictable. Or

```
: WHISTLE BEGIN CLICK AGAIN ;
```

(but only at the end of the session - why?). Or

```
: SQUEAK 0 1000 DO 1 PAUSE CLICK -1
+LOOP ;
```

etc. probably ad nauseam.

You can work in whatever number-base you like. There have been numerous BASIC programs published in the various computer magazines (probably at least \$0014 of them) to convert numbers from hexadecimal to decimal or vice versa, but in Forth it is very easy:

```
DECIMAL 120 HEX .
```

will give you '78 OK', where \$78 is the equivalent of the decimal number 120. [DECIMAL] sets the number-base to ten, and in nearly all Forth versions [HEX] sets it to sixteen. If you have a lot of conversions to do, you can define a word

```
: TOHEX BASE @ HEX SWAP 0 D. BASE ! ;
120 TOHEX
```

This will give you '78 OK' if you started off in base ten, but if you tried it immediately after the previous example it gave you '120 OK' because you were already in base sixteen. What happens is this: [BASE @] gets the present number-base, [HEX] sets the base to sixteen, SWAP lets you get at the number you wanted to convert, and [0 D.] prints it as an unsigned number in the range 0-FFFF. If you wanted a signed number from -8000 to +7FFF you could use simply [.] instead. Finally [BASE !] restores the old value of number-base. This word will now convert from any number-base that you care

to use into hexadecimal.

The two standard number-bases are base ten and base sixteen; but supposing that you want to use base two, perhaps for making shape tables or for displaying the processor status register or the state of an eight-bit input port, you can define

```
: BINARY 2 BASE ! ;
```

and now [BINARY] will change the number-base to two. So

```
DECIMAL 99 DUP HEX . BINARY .
```

will give '63 1100011 OK', in other words it takes the decimal number 99, makes a DUPLICATE copy of it, and displays it first in hexadecimal and then in binary. It leaves you in binary, so that typing

```
99
```

will give '99 ? MSG # 0', which means 'unknown word', since [99] is neither a Forth word nor a number in base two.

It is easy to develop simple control programs or 'applications' as they are known in Forth. If for instance there is a Digitalker card in slot 2 of the APPLE, the following makes it produce its complete range of words and noises:

```
HEX
: TALK 90 0 DO 1 C0A0 C!
BEGIN C0A0 C@ 2 MOD UNTIL LOOP ;
TALK
```

That counts from 0 up to \$8F, the range of values that Digitalker translates. It puts each value in turn into the Digitalker output register at \$C0A0 (\$C0B0 for slot 3, etc.), which starts the noise. It then enters a loop which scans the Digitalker status register (also at \$C0A0) until the lowest bit becomes a '1', which indicates that the noise has been completed and it can go on to the next one.

The current disk version of Kuntze's '4TH APPLE!' is available from the software library and includes demonstration screens on noises, a simple drawing application, the fig-Forth screen editor (use with care), a compiler which handles most 'normal' words, a few utility words, an assembler which is licenced for educational distribution but which is insecure, and a special vocabulary which allows Forth to read and display the time using a Glanmire Electronics Micro-Watch. It is public domain

software, meaning that you may copy it and distribute it further if you wish (RUN NOTES first for a notice to this effect) at your own risk! There is one known bug in the current issue: DR0 and DR1 do not do exactly what one would expect... also be warned that pressing reset at present kills DOS. If this matters (FORTH only uses RWT5 itself) you can do

```
MON
*3EAG      (reconnect DOS)
*6104G     (warmstart FORTH)
```

If you have the disk,

```
12 67 INDEX
```

will give you an idea of what screens are available;

```
12 LIST
```

will display the contents of screen 12, while

```
12 LOAD
```

will actually carry out the instructions on that screen. All screen transactions should be done in DECIMAL.

Before you do anything, make a copy of the disk using the COPYA or COPY program from your system master disk. The disk VTOC has been fixed so that all tracks up to track 17 (the CATALOG and VTOC track) are marked as used, so that any Forth screens (12 to 67) will be copied along with DOS and with the normal DOS files that occupy the latter half of the disk. I can supply an editor manual if needed - no room on the disk.

For those who want a more advanced package there are at least five other Forth implementations for the APPLE: Kuntze has an extended version, Cap'n Software has one, Transforth is extended in various directions including maths, Metacrafts is powerful but I don't know much about it yet, and GraFORTH is a very powerful graphics and music language which supports recursion and allows the creation of 'autorun' packages but lacks some of the flexibility of the others (but it has to be seen and heard... and it is not too expensive either).

For those with other fig-FORTH systems, the only thing above which will need changing (apart from the Digitalker section) is CLICK. Refer to your system manual for information on how to click the speaker; you will probably have to TOGGLE one bit of an output port.

User Hostility

Why User Friendly?

by Bob Raikes,

Don't those patient, ever calm people who never get excited or angry drive you mad? They do me.

How much worse it is when a dumb machine is the same. Such patience. However many times you put the data disk in instead of the program disk, that flaming machine simply beeps, displays 'Insert Program disk and hit any key to go on' and waits patiently. Absolutely maddening.

I would like to make a plea for user hostile programs. No more endless prompts and quiet patience. How about the sound of a raspberry, followed by:-

"Are you stupid or something? I said the PROGRAM disk, you idiot".

Then if you repeat the mistake:-

"I've told you once. Put the P R O G R A M disk in now. Come on, I'm waiting".

If you make the mistake again, the program would initialise the disk, leaving only a file called 'SERVES YOU RIGHT' and a screen display of 'DAMN HUMANS'.

No more of this simple prompt such as a ? Replace this with 'WELL??' and the sound of a tapping foot.

Some programmers have come to these same conclusions before me I am sure, and have worked on some very subtle ploys. The most user hostile program I have met is an old version of Computer Ambush. After 10 hours playing (25 mins per turn!), it crashed and destroyed the game when I tried to save it with 'DISK FULL ERROR'. Anybody got any other nominations?

Quick Tip

by R. C. Lowe



CONTROL CODE SPOTTING

If you want to know if the name of a file on disk contains a control code try CATALOGing while in INVERSE mode, it won't tell you what the control code is or where it is but it will tell you how many (if any) are in the name.


Small Ads

For Sale:

48k ITT 2020, Applesoft in ROM, two ITT disk drives with controller, Ferguson 14" B&W TV, Parallel printer card with additional circuit to drive V24 serial, Transdata 300 thermal printer with keyboard, paddles and games. £500.



Contact: Alan Duerden,  

Apple II+ 64k, two disks, colour card, Centronics parallel card, paddles, docs, etc. £800.

Contact: Derek Page on  during office hours.

Apple II disk drive + controller £160
APS 9" monitor £25

Various games in mint condition all at half price.

Ring David Girdler on   during office hours.

Neptune 64k Extended 80-column card for Apple II/e. £100 o.n.o.





Ring Brian Poulton on 

EDUCATIONAL/GAMES SOFTWARE

Approximately 25 packages, used once in original boxes with all documentation normally only for sale in the United States and worth about £700

need Apple II, II+, Applesoft 48k, DOS 3.3 //e, disk drive or Franklin Ace, disk drive

£100 the lot or will split

Contact: James Nalty,  
 

Mathematicians do it with tables.

Software Engineers do it in bits.

Software Review

Apple Assembly Language Course

Price: £14.50

Suppliers: Honeyfold Software Ltd.

Standfist House, Bath Place

High Street, Barnet, London EN5 1ED

Reviewed by Seth Proctor

This "course" comes in a video-cassette type box, consisting of a 5 1/4 inch disk and book. The disk holds the assembler, which may be used on any Apple I or on a IIe, and a "tutorial"; the book, meanwhile, explains the codes of the Apple's processor chip - the 6502 - with various appendices having useful tables which are vital to programming. But what is "assembly language" and the "6502"?

ASSEMBLY LANGUAGE.

As most of you may be aware the Apple comes with a standard language - BASIC, or Applesoft. The main advantage of this language, believe it or believe it not, is that it is easy to read and to understand with its instructions being rather like English, and therefore relatively easy to use. So to print something one would key in "PRINT..."; to go in a loop ten times

```
FOR X = 1 TO 10 ... NEXT X
```

and so forth. However, this is not the language of the computer, for it is, putting it very simply, a collection of switches (which may be on or off) with those which run and control the Apple on a special kind of chip - a processor chip called the 6502 - and everything has to be scaled down to this level. So, for example, BASIC programs are decoded into a format which the computer can understand - that is machine code. So, obviously, it would be better to write programs in machine code to cut out the decoding. Therefore why not write programs in this code? Because it is a laborious task whereby everything, as I said earlier, has to be scaled down, or decoded, by the programmer before writing the program. Even a simple PRINT statement requires a lot of work, so a compromise has to be reached between legible languages like BASIC and the illegible machine code. This is assembly language, whose prime advantage is speed, for there is a closer relationship between assembly language and machine code than between BASIC and machine code. If one, therefore, wishes to write programs where

speed is the name of the game (e.g. animation or fast sorting), then one has to purchase an assembler which allows a user to write programs in assembly language and to decode them into machine code. The Apple Assembly Language Course by Malcolm Whapshott in the Dr Watson series is one which is available and under review now. This review - done with a IIe - is split up to cover the package's four components - the course, the tutorial, the assembler and the package itself.

THE COURSE.

This makes the assumption that those reading the book have no prior knowledge of assembly language, so a "softly-softly" approach is given and maintained throughout. Thus one starts with an introductory chapter on understanding the language, and machine code, as well as understanding and writing instructions. Thence to the introduction and on through the various chapters of more concepts which are needed to write programs, including the necessary understanding of binary and hexadecimal numbers, so that by the final chapter one should be proficient in writing machine code programs with the assembler and knowledgeable about the built-in subroutines on the 6502 chip. Whether the course is fast or slow, is debateable. I will admit that I found it slow, but I must point out that I have done work in assembly language (albeit three years ago on an ICL mainframe). However, I maintain that slowness is nothing like as bad as a fast breeze through, because the subject is quite complex if you are a complete novice at working in machine code/assembly language - you need to take your time and not get dazzled and overcome by computer science.

THE TUTORIAL.

The disk tutorial consists of two main parts - questions & answers and a "counter" facility. The questions mainly ask the user to calculate the equivalents of given numbers in various bases - for example binary coded decimal to decimal, decimal to hexadecimal, etc., though not, say, binary to hexadecimal, which may be needed by some users. These are in addition to the descriptive text and a few questions & answers in the book. As regards the "counter" facility, this is a very rudimentary, basic facility which has rows of numbers in boxes which are in two sets - decimal, binary & binary coded decimal, and decimal, binary & hexadecimal. The object of the exercise is to see what happens if one

RAMVIEW

This is Elite's own 80-column card for the Apple //e. It is completely compatible with Apple's own 80-column card. The only differences are the price and the fact that our board can be upgraded to be a 64K/80-column card by simply plugging in 8 chips.

RAMVIEW may also be purchased from Elite Software as an upgraded 64K board.

RAMVIEW is supplied with an instruction manual, is fully tested before despatch and is backed by Elite Software's **full one year's warranty**.

The price of a **RAMVIEW** is **£60 + VAT (£69)**.

The price of a **64K RAMVIEW** is **£120 + VAT (£138)**.

Postage and packing is **free!** *

Dealer enquiries welcomed.

Apple is a trademark of Apple Computer Inc.

'A'dds or 'S'ubtracts from the decimal number, which, incidentally, may be started between 0 and 99 with the first set of boxes and between 0 and 255 for the second. Methinks it was unwise of the programmer not to allow negative numbers in the boxes. That is to say if one has a 0 display then when 1 is subtracted the display goes to either 99 or 255 with the appropriate equivalents given. This could be confusing, I hope that this will be changed.

THE ASSEMBLER.

The course's assembler is very adequate and allows the user to enter, list and run assembly language/machine code programs, as well as to insert spaces and to move code. Subject to my notes in "The Package" below I must admit to being disappointed with the assembler for various reasons:-

1) When entering assembly code labels may be used to refer to a particular section of memory. However, upon leaving this facility, the assembler changes these into their memory locations. So, for example, if a section of code is called LOOP1 and a screen location defined as POSITION1 then the entry may look like

```
LOOP1 STA POSITION1
      |
      v
JMP LOOP1
```

However this could be changed to

```
773 STA 1024
      |
      v
JMP 773
```

This, I feel, defeats the main purpose of using such labels - to read the listing easily without necessarily having to refer to peripheral codes or text. True, they help the programmer in writing the code, but how about reading it some time afterwards? This is especially so with large programs which use many subroutines. For example, one may remember that POSITION1 is the first screen location, but not 1024, and so forth.

2) My second disappointment was that to insert code, say omitted in the entry facility, the programmer has to first calculate how many bytes are needed, then insert the appropriate space using the insert facility, thence back to the entry facility to enter the relevant code. All very cumbersome and time consuming, and I

certainly found that for small programs it is better to start from scratch. An 'add' facility would have been far better.

3) The assembler does not allow the user to have comments with the entered code, which is a great shame as machine code/assembly language programs are sometimes long and difficult to follow. (Having said that please see "The Package" below).

These apart, as I said, the assembler is okay and has added facilities, as one would expect, to save and load programs (though only using drive 1) and to go in and out of monitor. The only problem I encountered using these was that they were not described in a clearly marked section, which in the event was fairly late in the book; it may have been better to have a 'help' facility on disk (to be printed to screen or paper) or, as I said, a clearly marked section in the book.

THE PACKAGE.

I must confess that I was disappointed with the overall packaging of the course. In the book there were a great number of misprints and dropped lines (normally after subscripts like 2 and 10). Also lower-case inputs are not possible despite there being two assemblers - one each for the I and II. This, I feel, devalues the course as many IIe products recognize the existence of lower-case characters and cater accordingly. Another problem encountered was the fact that only drive 1 may be used, which is fine until one exits the assembler and subsequently wishes to re-RUN it. It needs a program on the course disk in drive 1, otherwise the assembler program crashes. In short one has to go through the irksome task of ensuring that the right disk is in drive 1 - the data disk when loading/saving programs or the course disk when wishing to RUN the tutor or assembler. Another comment to note is that the book is not spiral-bound - unlike Apple manuals - which may be disconcerting for some users; I am told that this would raise the price to an unacceptable level, so this is fair enough. Furthermore there are no reference cards supplied - these, I feel, would be of great benefit as at some stages of the course I was referring to a program/exercise, the solution, the Apple character-set and the Apple screen locations. For ease of referral the last two could be given on cards as well as in the given appendices. (Such "cards" need not necessarily be expensive stiff card, a pull-out section could do).

However, I have reviewed the First Edition of the course, and by the time this review is printed a second edition should be for sale (or be imminent). This has, so I am promised, no misprints or dropped lines and does allow for comments in the assembler. As regards my other comments these are being considered for future editions. In short the publishers are very much aware of the need to continually improve their wares, so the second edition should be far better than the first - if so then the product has been greatly uplifted and should not collect dust on seller's shelves. (A note to our continental members: The second edition, unlike the first, will be available on the continent. Questions of where, please, to the publishers).

SUMMARY.

On the whole the course is good value-for-money, and is an inexpensive introduction to the complexities of assembly language. However, I would imagine a would-be serious programmer may have to invest in a more versatile assembler which would give the useful aids mentioned above, although not only would it be wise to see the latest edition before looking elsewhere but it may be best to buy this course in any case to make sure that assembly language is for you.

Mac Note

by P. Knight.

MICROSOFT MULTIPLAN FOR APPLE MAC.

For: Very powerful when "opened".
63 columns wide (10 char. cols.).
255 rows.
Editing quite good.
Display very good.

Against: On opening, displays keyboard of US machines. Thus typing on bottom line of UK keyboard produces characters displaced one key to right AND Return key (L-shaped on UK board) and backspace and spacebar do other things!

Also for US a "dollar" command will globally label all numbers in selected column. No such choice for "£".

How to SUM? Not much good in M'PLAN manual. A useful guide which one can understand will be found in Cary Lu's "The Apple Mac Book" - an excellent book quite without parallel.

Lastly, why does Microsoft UK documentation

not include supply of a backup disk. It costs £25 for a replacement disk outside warranty. This is the charge also levied by Microsoft UK for a backup disk! Surely this is a not-clearly-displayed cost which comes under the Trade Descriptions Act and of interest to Trading Standards Officers. However, USA documentation (left in box "in error" according to UK office) tells you how to get it for \$10.

Of course, you're still stuck with US keyboard - no £ sign, etc. Who released this package in UK? The program also does not recognise dual disk drives - asks for Master disk when it is already loaded in other drive.

In returning package to P & P I said that we could alter key caps but surely that would infringe disk warranty, also that I didn't know how to amend program for 2 disk drives even if it were possible.

Where can I get a reliable UK-style spreadsheet for Mac which is anglicised for 2 drives? I have hobby and business interests waiting for me to erect a working system about 18 cols. by 250 rows (10 char. cols). I also badly want a Tele/Address package plus 3 - 4 option boxes so that I can sort on name, company, telephone number, etc.

I suggest Microsoft (particularly UK end) should "pull up their socks" and read recent criticism in US press of their "Word" and lack of a Hotline.

I have just rung Apple USA and they say they are not doing a Mac/M'Plan despite their C'LOE.DMO.400 which I ordered.

BEWARE!

There are disk drives appearing on the market which use metal positioning bands to move the head. On an Apple disk drive the track positioning is considerably less accurate so in order to ensure consistent booting track 0 tends to be fairly wide. This means that a disk initialised or copied on a metal band type of disk drive may prove difficult to boot on a standard Apple drive. The converse does not apply.

Computer Operators do it in shifts.

Database programmers do it with relations.

IMPORTANT NEWS FOR APPLEWRITER USERS!

By Dick Menhinick (with a little help from Apple UK!)

The following instructions if carried out accurately, will allow your copy of Applewriter //e to be modified to enable the NUL character (Decimal 0) to be sent to your printer.

NUL cannot normally be sent from Applewriter since the NUL code is not accessible from the keyboard. Unfortunately, NUL is needed in many printers (Epson FX80, Taxan KP810) to activate functions such as underscore, super and subscript, proportional spacing etc. In the case of the Canon FW1080 and Taxan KP810 near letter quality (NLQ) printers the absence of NUL is particularly irritating since it without it, the second (optional) NLQ typeface cannot be selected (there are 14 different NLQ typefaces available for the Taxan KP810 now!).

The following instructions apply only to Applewriter //e. They modify the program to enable the 'Delete' key on the Apple//e produce a NUL code when in 'Control-V' mode. Delete works normally unless Control-V has been pressed. (This has been printed on a Taxan KP810 with the standard and *italic* NLQ typefaces, using Applewriter //e)

General instructions are in this typeface.
You type the instructions printed in this typeface

1. Boot your DOS 3.3 master disk
2. Remove the write protect label from your Applewriter //e disk. Put the Applewriter //e disk in your disk drive and close the drive door.

3. Now type the following lines, each followed by the RETURN key.

```
BLOAD APWRT][E
CALL -151
1CB1:EA EA EA EA
3EBA:EA EA EA EA
3DOG
UNLOCK APWRT][E
BSAVE APWRT][E;A$1900,L$2F56
LOCK APWRT][E
```

```
BLOAD APWRT][F
CALL -151
1D81:EA EA EA EA
4033:EA EA EA EA
3DOG
UNLOCK APWRT][F
BSAVE APWRT][F;A$1900,L$30D1
LOCK APWRT][F
```

That completes the modification. Fit a new write-protect tab to your Applewriter diskette.

You should now Boot the Applewriter disk and give it a try! Remember, to get the NUL character from the Delete key you will need to have pressed Control-V first, but as it will be part of a longer command string for the printer you would normally have done that anyway!

Now, a short demonstration of what can be done with a Taxan KP810 now that we can switch typefaces!

Taxan Plain typeface

Taxan Outline typeface

Taxan Gothic typeface

Taxan HPEEK tpeEζayE!

Taxan Handwriting!

Taxan KP801 d typeface

Taxan Tall typeface

Epson DX100

The Epson DX100 Printer and Keyboard.

A review by J. W. Panks.

Epson have for some years been connected with dot matrix printers. These have been of good quality and reliable. They were probably the first company to introduce printers at a price that most users could afford. In recent months Epson have announced many new products. These have included computers, printers and plotters. One product that has been added to the range is a Daisy Wheel Printer called the DX100. This has been overshadowed by the other products and has not been widely advertised.

The Epson DX100 is in fact a Brother-manufactured printer similar to a Brother HR15. It is designed to be a low cost, high quality printer with dual use as a computer printer or a typewriter. It is available with either a Centronics or RS232C interface.

The printer is built on a chassis which appears to be very strong and is encased in plastic which is a similar colour to the Apple beige. The top lifts up to provide easy access to the ribbon and daisy wheel. The print head is pulled across the platen by the usual wire cable type mechanism. It appears to be capable of this job and should prove reliable. The electronics are at the rear of the machine under the platen. The rear panel has D-type plugs for the printer interface and for the optional keyboard. Also provided is a DIN type socket for the optional sheet feeder. Dip switches have usually been hidden inside the machine; on the Epson they are placed on the rear panel, which makes changes to them very easy. There are two provided. The first selects the language of the daisywheel; the second is used to set page lengths. They are well documented in the manual. On the front of the machine is a group of switches that allow Line Feed, Top of Form, Copy, Selection of Line and Pitch settings.

The printer is designed for medium use, and would be ideal in the home or for light office use. The print speed is slow in comparison with other more expensive daisy wheels or dot matrix printers. I have timed various print times for text and the average speed is 13 characters per second. The print head is bidirectional and coupled with the 3k buffer provided as standard makes the printing of single pages fairly quick. Longer

articles take time and if you need the computer back whilst long articles are printed you will need to acquire a large buffer.

The noise is acceptable providing the printer is positioned on a surface that does not vibrate, and is solid. I placed the printer on a fairly shaky table and the machine nearly threw itself on the floor. It does tend to shake, although putting some foam underneath cuts the noise and shake in half.

The features of the machine are very good and are not found on some that cost well over £1000. It will print at 10, 12 or 15 characters per inch with Proportional spacing if required. The appropriate daisy wheel will be needed to get the best from the different pitches. Double strike, shadow printing and underline are also available. Two colour printing can be done when the printer is used with a computer; when a keyboard is used the red print becomes a corrector providing you change the small ribbon.

The machine has a very useful copy facility which uses the 3k buffer to copy documents without the computer being tied up. I found this useful when writing similar letters to many people, and I think that it is cheaper than using a photocopier. The letter is loaded into the printer buffer and then you use the copy key on the printer to print it out. If you run a small business this would be useful coupled with the optional sheet feeder.

The printer has a useful top of form key which allows you to load single sheets to a pre-determined position by simply placing the paper and pressing the TOF button. The next time you press the button it ejects the paper. Other commands that can be entered in off line mode are line spacing and pitch. These can also be set by software from within programs.

Other features include auto underline, auto strike out, super- and subscript, half line feed in both directions and the setting of print head movement to 1/120th of an inch. Line spacing can be set to 6, 4 or 3 lines per inch by using the front panel or from software. The line spacing can be set in increments of 1/48th of an inch from software, which could make it easy to use the printer for graphics.

The Epson is compatible with Wordstar and uses standard Diablo protocols. I have used

it with Applewriter II, Format 80, Letter Perfect, Wordstar and Zardax with no problems. If you wish to use it in Proportional Mode you will have to configure your program to the printer. On some this is easy, on others it may take some time. If you really need proper proportional spacing you would be wise to get advice before buying any particular program.

There is an optional extra 48 character keyboard for the Epson which turns the DX100 into a super typewriter with all the latest fancy electronic typewriter facilities, including auto correction. The keyboard plugs into the rear of the printer and can remain there when the printer is being run by a computer; there is an on/off switch on the keyboard. The keyboard is a standard QWERTY keyboard with optional keyboard overlays selected from a switch. This allows the use of different daisy wheels. Other features included are pitch and line spacing switches. The keyboard has adjustable feet for angle and this can be set at 15, 10 or 5 degrees. Some of the keys not normally found on a keyboard are super- and subscript keys, correction and relocation keys (used for auto correction), express return key (and back space) and a code key. The code key allows the setting of decimal tabs and line indents.

The keyboard allows automatic correction in the line that you are typing and it also has a 48 character buffer. The keyboard is ideal for those that like myself find setting up the computer for text entry a waste of time when only a small letter or label is required. It also has the tendency to stop the wife moaning about spending more money; she understands a typewriter but is not interested in learning the word processor. That is the only problem with the keyboard. My wife constantly outruns the buffer. It would appear that the printer can handle data from the computer quicker than the keyboard. This has the tendency to cause major problems with fast typists. They find it hard to slow down to the speed of the print head and as they rarely look at the keys or the typing they find that they have lost many characters. However I find the keyboard excellent but I only type at 20 - 30 words per minute.

As I said previously the machine is really a Brother in disguise and therefore Brother ribbons and daisy wheels are required. There are a fair number of daisy wheels available for the DX100 but they are fairly expensive at roughly £18 each. Therefore if you need

plenty of different types you should bear in mind the cost of the daisy wheels. Printer ribbons are better priced and you can get one for between £3 to £4 depending on the supplier. You can get Single Strike ribbons which produce perfect copy although they don't last long and Multi Strike which produces reasonable copy but lasts five times as long as single strike. Nylon Ribbons which are similar to normal typewriter ribbons in a cassette are available but the print quality is only fair and quickly becomes poor after use.

The documentation that accompanied the printer was adequate without being too technical. The machine tested was a sample machine and the actual documents supplied may be improved on the production model.

Other add-ons include a sheet feeder and a tractor unit. These are fairly cheap compared to other printers. I have not tried them out but if the quality is as good as the printer then they would be a good buy, providing you can justify their use.

My conclusions are that this printer is a high quality printer at a reasonable price. The speed is not very impressive but for a fast daisywheel you will pay a lot more. If you want top quality, coupled with medium use, without paying a fortune, this printer is for you.

I must thank EPSON (UK) and in particular John Sharp for allowing me to use the printer. It has been enjoyable and I will now be able to produce an Applewriter Glossary file for the Epson DX100.

IMPORTANT

PLEASE NOTE

There are two major telephone number changes for BASUG. Please alter these numbers in your books (or wherever you keep them). These numbers are valid as from NOW.

ADMINISTRATION (Fran Teo)

Newbury ((116373) 463365)

BULLETIN BOARD (Tony Game)

Felixstowe ((116373) 463365)

DERING PC2000 A new concept

64K RAM, twin disk drives
£795

Complete business system
with software, from
£1236

- Runs BASIC, CP/M, PASCAL
- Multi-operating systems
- Large software range
- Twin built-in disk drives
- 64K RAM expandable to 192K
- Separate low profile keyboard
- Built-in printer port
- Full colour-RGB, composite
- High resolution monochrome
- 10 MByte hard disk available
- Fully guaranteed



DERING SYSTEMS
Unit 22C, Low MHI
Dewsbury, WF13 3LX
Phone (0924) 499366, ext. CC
Telex 83147 VIA OR,
DERING

This new economy priced system utilises universally available software, comprising many thousands of programs including CP/M. The addition of a monitor and a printer (all that is required for a complete system. Prices start at £795.00 for the PC2000. A complete business system with Microledger accounting software, printer and soft amber screen monitor would cost £1236.00. The PC2000 with general purpose business software comprising a spreadsheet, the 80 column word processor FORMAT 80 and ACCESS database - £1029.00. A top-of-the-range system including a 10 MByte hard drive, daisywheel printer, monitor and universal software pack - spreadsheet, word processor, database and accounting system would be £2581.00.

This soundly built system is backed by the service and reputation of an electronics company of 24 years standing.

Quiz Writing

Writing A Quiz Program In BASIC

by Hugh Dobbs, Newtown School, Waterford

A number of my students have been seen expending great efforts on writing quiz programs to baffle their friends; I was alarmed to see how much of the work was unnecessary, and for the past few weeks we have had an intensive course on the subject. As observed, a typical program would run like this:

```
10 PRINT "WHAT IS THE CAPITAL OF
FRANCE"
20 INPUT A$
30 IF A$ = "PASS" THEN GOTO 60
40 IF A$ <> "PARIS" THEN GOTO 10
50 PRINT "RIGHT"
60 PRINT "WHAT IS 32 + 45"
70 INPUT A$
80 IF A$ = "PASS" THEN GOTO 110
90 IF A$ <> "77" THEN GOTO 70
100 PRINT "RIGHT"
110 .... ETC....
```

... I have of course removed numerous syntax errors and standardised the line numbers, but the overall structure is as shown. A little calculation shows that for a 100-question quiz one would need 500 lines of BASIC code, including 200 GOTO statements, any one of which might be to a wrong line number -- for example, did you notice that line 90 there should end with 'GOTO 60' so that the question gets asked again? I call that the Hard-Work Version (HWV).

Clearly it would be much better if all that testing and branching could be done just once; so I introduce the Subroutine Version (SV):

```
10 GOTO 1000
100 PRINT Q$;
110 INPUT A$
120 IF A$ = "PASS" THEN GOTO 150
130 IF A$ <> R$ THEN GOTO 100
140 PRINT "RIGHT"
150 RETURN
1000 LET Q$ = "WHAT IS THE CAPITAL
OF SPAIN"
1010 LET R$ = "MADRID"
1020 GOSUB 100
1030 LET Q$ = "WHAT IS 5 TIMES 7"
1040 LET R$ = "34"
1050 GOSUB 100
1060 ....ETC....
```

Calculation here indicates that for a 100-question quiz you need 307 lines of BASIC, with a total of 3 GOTOs; and whereas you have 100 GOSUBs they are all identical so that any mistyping should be obvious. The subroutine is placed ahead of the main program for two reasons: first, most versions of BASIC search through from the start of a program for the line number addressed by a GOTO or GOSUB (which takes longer for later lines in the program), whereas the RETURN already has the absolute address of the GOSUB which called it on the stack and does not have to search for it; and second, when you LIST the program the bit that does the work is displayed first, which is almost certainly what you want.

The difference between these two versions becomes more obvious when you get dissatisfied with the way the program works at present and want to add facilities to it. One obvious deficiency is the lack of scoring. You decide to keep a score, which means setting up a counter and adding one to it for every right answer, and then printing the score at the end. On the HWV this involves adding the following lines:

```
5 LET S = 0
55 LET S = S + 1
105 LET S = S + 1
155 ....ETC...
9998 PRINT "YOUR FINAL SCORE IS "; S
9999 END
```

... for a total of 603 lines so far; the SV only requires:

```
5 LET S = 0
145 LET S = S + 1
9998 PRINT "YOUR FINAL SCORE IS "; S
9999 END
```

... for a total of 311 lines. But perhaps you would like the score to be printed every time you get the answer right; to do this with the HWV:

```
57 PRINT "YOUR SCORE SO FAR IS "; S
107 PRINT "YOUR SCORE SO FAR IS "; S
157 ....ETC....
```

... only 99 of these, making 702; while for the SV:

```
147 PRINT "YOUR SCORE SO FAR IS "; S
```

... suffices, making 312 in all.

This looks pretty good, but further savings are still possible depending on the

abilities of your dialect of BASIC. So far I have been assuming a fairly minimal Microsoft BASIC, and the two programs will run with minor changes on most machines.

Some BASICs such as Sinclair and Atom BASICs allow a 'computed GOSUB' — that is, they allow 'GOSUB <expression>' as well as 'GOSUB <line number>'. (For the Apple use ON...GOSUB or one of the & extensions to BASIC). With these the whole program can be inverted:

```
10 LET S = 0
20 LET Q = 0
30 GOSUB 1000 + Q * 30
40 IF Q$ <> "END" THEN GOTO 70
50 PRINT "YOUR FINAL SCORE IS "; S;
  " OUT OF "; Q
60 END
70 LET Q = Q + 1
80 PRINT "QUESTION "; Q
90 PRINT Q$;
100 INPUT A$
110 IF A$ = "PASS" THEN GOTO 30
120 IF A$ <> R$ THEN GOTO 90
130 PRINT "RIGHT"
140 LET S = S + 1
150 PRINT "YOUR SCORE SO FAR IS "; S;
  "OUT OF "; Q
160 GOTO 30
1000 LET Q$ = "HOW MANY CORRECT ANSWERS
  ARE ALLOWED FOR IN THIS TEST"
1010 LET R$ = "JUST ONE"
1020 RETURN
1030 LET Q$ = "WHAT IF THE COMPUTER'S
  ANSWER IS WRONG"
1040 LET R$ = "TOUGH"
1050 RETURN
1060 ...ETC...
4000 LET Q$ = "END"
4010 RETURN
```

This version has a total of 318 lines, and will run more slowly than the SV, but it has an advantage if you are very short of memory space: 'GOSUB 100' will take up more memory than 'RETURN': Sinclair, 5 bytes as against 1 (I think); Atom 5 bytes ('GOSUB') as against 2 ('R'); thus saving 400 or 300 bytes on the latter part which more than makes up for the extra lines. Of course both BASICs mentioned require further lines for dimensioning string variables, etc. and the Atom will refer to SQ rather than Q\$ throughout.

Other BASICs have READ, DATA etc., which allow of further improvement:

```
10 LET S = 0
20 LET Q = 0
```

```
30 READ Q$, R$
40 IF Q$ <> "END" THEN GOTO 70
50 PRINT "YOUR FINAL SCORE IS "; S;
  " OUT OF "; Q
60 END
70 LET Q = Q + 1
80 PRINT "QUESTION "; Q
90 PRINT Q$;
100 INPUT A$
110 IF A$ = "PASS" THEN GOTO 30
120 IF A$ <> R$ THEN GOTO 90
130 PRINT "RIGHT."
140 LET S = S + 1
150 PRINT "YOUR SCORE SO FAR IS "; S
160 GOTO 30
1000 DATA "IS THERE ANYTHING TO BE
  GAINED BY USING DATA STATEMENTS",
  "YES"
1010 DATA "WHAT IS THAT","BREVITY"
1020 ....ETC....
9999 DATA "END","PHEW"
```

That brings the total down to 117 lines, including the printing of the question number which was not part of the first two versions. Further space can be saved by packing more than two DATA items per line, but I don't recommend it until the program is fully debugged as the layout used makes it very easy to see which question has which answer, and to check question number Q you simply work out $Q*10+990$ and LIST that line: Most versions of BASIC will allow you to leave out the quotation marks around the DATA items, but this can give rise to invisible errors as follows: if you type

```
1050 DATA what is 7 + 8, 15
```

and press the space bar before pressing <return>, the instruction READ Q\$, R\$ will pick up what you would expect for Q\$, but R\$ will be the three-character string "15", and the victim who types "15" in response will be marked wrong. Incidentally the reverse may also apply: if the correct answer is "15" then "15" will be wrong. This does not usually apply to spaces at the start of a string... Anyway, only remove the quotation marks after the program is working properly, and only then if you are short of space for the later questions. The invisible errors can be brought out in APPLESOFT by using INVERSE before LISTing the DATA section; they can often be produced by careless editing.

Finally, perhaps your computer has disks. A quiz program is an example of serial data processing, and can easily be rewritten to take its input from text files on disk. Since

all the disk operating systems I have met have been different (!) I shall just give an example using APPLESOFT and DOS 3.2 or 3.3...:

```

10 LET Q = 0
20 LET S = 0
30 LET DS = CHR$(4)
40 PRINT "WHICH QUIZ WOULD YOU LIKE ";
50 INPUT FS
60 PRINT DS; "OPEN"; FS
70 PRINT DS; "READ"; FS
80 INPUT QS
90 IF QS <> "END" THEN GOTO 120
100 PRINT DS; "CLOSE"
110 PRINT "YOUR FINAL SCORE IS "; S;
    " OUT OF "; Q
115 END
120 INPUT RS
130 LET Q = Q + 1
140 PRINT DS
150 PRINT "QUESTION NUMBER "; Q
160 PRINT QS;
170 INPUT AS
180 IF AS = "PASS" THEN GOTO 70
190 IF AS <> RS THEN GOTO 150
200 LET S = S + 1
210 PRINT "RIGHT. YOUR SCORE SO FAR
    IS "; S
220 GOTO 70

```

This takes only 23 lines, but of course you still have to write the questions and answers. To do this you can use any suitable text-editor, such as the APPLE DOS Toolkit editor (BRUN EDASM). Type in questions and answers alternately, finishing up with END in place of a question:

```

WHAT IS BLACK AND GREEN AND READ ALL
OVER
A MONITOR
WHAT IS THE SOUL OF WIT
BREVITY
END

```

Obviously this program can be used with any number of different quizzes, simply by writing different text files to go with it, so it is worth while building as many improvements into it as you can. Perhaps you would like the screen cleared before each question... perhaps you would like unnecessary spaces removed from answers input... perhaps you would like the program to present a menu of the quizzes on the disk and allow the victim to choose one by simply pressing a key... perhaps the victim should only be allowed three attempts at a question... perhaps the victim should be allowed to escape without attempting all 100 questions... it's up to you.

String Print

by R. C. Lowe

In response to Word Wrap by Tony Game in the August Hardcore, here is a shorter faster routine that does the same thing. When compressed, it only uses a few lines. I have altered this routine that I have been using for some time so that it is compatible with Word Wrap.

The only problem that I have found with it is that, as it is recursive, it will run out of stack space if the string to be printed is too long or if the line length is set too short.

```

30 REM STRING PRINT NS
40 IF LEFT$(NS,1) = " " THEN NS =
    RIGHT$(NS, LEN(NS) - 1): GOTO 40
50 IF LEN(NS) < PL THEN PRINT NS:
    RETURN
60 S = PL
70 FOR PS = PL TO 1 STEP - 1
80 IF MID$(NS,PS,1) = " " THEN S
    = PS:PS = 1
90 NEXT PS
100 PRINT LEFT$(NS,S):
110 NS = RIGHT$(NS, LEN(NS) - S)
120 GOSUB 40
130 RETURN

```

Software Library

by Jim Panks

SOFTWARE GROUP.

All 3.2 software disks have now been distributed and when they are returned the 3.3 will be sent out. If you have ANY difficulties ring your respective manager, the address and phone number is in the manual supplied.

Also:- Any members wishing to help in the software library should get in touch with the software group manager at the P.O. Box or ring the number on the back of the membership card. Although we have quite a few helpers we will not turn anyone away. The more helpers the better.

NEW DISKS.

New Software Disks are being released. These will include a very fast disk copy program. Full details will be published prior to Christmas.

FORMAT-80

FOR THE APPLE][AND //e

Format-80 is simple to use. Text entry is as easy as using a typewriter. Editing and formatting is achieved with single key strokes. "D" for delete, "I" for insert, "J" for justify, "C" for centre, etc. Easy to remember commands because they make sense.

What you see is what you get. Format-80 performs virtually any editing and formatting function you can imagine, and displays on the screen the text exactly as it will print out.

It supports all Apple compatible printers.

Format-80 also includes a sophisticated mailing list, which comes complete with full sort and selection capabilities.

The program resides entirely in memory, including the mailing list. All drives are free for text and mailing list data.

It comes complete with a tutorial manual and a concise, easy to use reference manual, plus a handy user reference card, and is supplied on a standard DOS disc. The disc is not copy-protected.

System requirements:

64K APPLE][//e, and most Apple compatible machines.

Disc Drive.

80 Column Card. (supports most 80 column cards)

Also available as an upgrade to existing users of Format-80, including revisions to the reference manual, details of new features, and upgraded software.

Ramview: 80 column board for the //e, upgradeable to 64K.

PRICES **Format-80: £129.00**

Upgrade: £25.00

Ramview: £80.00

ELITE SOFTWARE COMPANY,

93 Eastworth Road, Chertsey, Surrey, KT16 8DX

Tel: 09328 67839

**PLEASE NOTE OUR
NEW ADDRESS**

Beginners' Page

MORE PEEKING AND POKING

By John Sharp

In the last HARDCORE we began to look at PEEKing and POKEing. Before continuing it is worth a few minutes to consider how to save yourself a lot of work when you have a list of POKES to enter into a program. If they are in a sequence as is most likely, why not let the computer do the work and use a loop to do it:-

```
10 FOR N = 768 TO 776
20 READ P
30 POKE N,P
40 NEXT
50 DATA 1,32,221,221,206,0,3,248,96
```

It will not have made much less work in this case but it will do so when there are many more POKES to do.

Now so far we have been mainly POKEing. How can we use PEEK in a program to let us do something constructive? One way might be to see if you have an APPLESOFT or PALSOFT machine, i.e. an APPLE or an ITT2020. One way is to look into a location you know to be different; Ian Trackman suggests location 62447. If you PEEK (62447) and get a number other than 0 then you are running on an Applesoft in ROM machine.

PEEKing is often useful to find out other things in programs. It could be used to find out if you are in FLASHING or INVERSE or NORMAL mode when you want to PRINT a new string. This uses the location we dealt with last time. For example:-

```
10 X = PEEK(50)
20 IF X = 255 THEN PRINT "YOU ARE IN
NORMAL MODE"
30 IF X = 127 THEN PRINT "YOU ARE
FLASHING AT ME!!"
40 IF X = 63 THEN PRINT "YOU ARE INVERSE"
```

This may not be a common occurrence but it could be useful if you were printing in different modes and wished to know which one you used last. Another case might be to test which key someone had pressed last. If you PEEK (-16384) then the value you get will be the ASCII value of the key pressed. This is illustrated by the following:-

```
10 FOR N = 0 TO 1000
20 X= PEEK( -16384)
```

```
30 IF X > 127 THEN PRINT CHR$(X);: POKE
-16368,0
40 NEXT N
```

The POKE -16368,0 is to reset the switch to read the next character typed in. You could use it to see which was the LAST CHARACTER typed on the keyboard, or indeed if any had been typed in whilst the program was running, without having to rely on the key being pressed at a specific time when the program requests it.

This suggests the following program for a simple reaction timer.

```
10 TEXT : HOME
20 FOR N = 1 TO RND (3) * 4000 : NEXT
30 PRINT "READY"
40 FOR N = 1 TO RND (3) * 4000 : NEXT
50 PRINT "STEADY"
60 FOR N = 1 TO RND (3) * 4000 : NEXT
70 PRINT "GO...."
80 POKE -16368,0
90 FOR N = 1 TO 300 : X= PEEK ( -16384) :IF
X > 127 THEN GOTO 110
100 NEXT
110 PRINT "YOUR SCORE WAS ";1000 - N ;
" CAN YOU MAKE IT LARGER ?"
120 END
```

A few comments would help those who do not understand. Lines 20, 40 and 60 put a random delay into the program between each statement. The resetting of the keyboard in line 80 stops you cheating by pressing a key before the "GO" comes up.

POKING THE HI-RES SWITCHES.

Have you ever tried to play a game and found instead of the correct graphics page the one from the graphics game you played before comes up. This is particularly common with the Integer games on the Software Distribution Library Disks. It means the switches are set wrongly for jumping between the various pages. These are POKE switches. If you read the APPLESOFT MANUAL it is confusing to say the least. An easier summary as follows is not foolproof because the switches are interdependent, but it usually works.

```
POKE -16304,0 ....TEXT TO GRAPHICS
POKE -16303,0 ....GRAPHICS TO TEXT
POKE -16302,0 ....GRAPHICS & TEXT TO
FULL GRAPHICS
POKE -16301,0 ....FULL GRAPHICS TO
GRAPHICS & TEXT
POKE -16300,0 ....PAGE 2 TO PAGE 1
POKE -16299,0 ....PAGE 1 TO PAGE 2
```

POKE -16298,0LO-RES SWITCH

POKE -16297,0HI-RES SWITCH

The last two are the most relevant ones if you have the trouble with being in the wrong type of graphics. Just press CTRL-C and type in the relevant POKE followed by a RETURN and rerun the program.

THE ESCAPE KEYS AND OTHER EDITING FACILITIES.

If you are a beginner you have probably glanced at the REFERENCE MANUAL and thought "Yes I'll look at that some other time, maybe in a few years time". Well there are some pages you might understand that are not in hieroglyphics. For example the pages on the ESC codes for moving the cursor on pages 34 and 35. Although most AUTOSTART ROM users are familiar with ESC IJKM they are not always aware of the ESC DBAC set. They give you similar movements, but with one difference. If you press ESC ABCD then the next key you press actually works. It is not a "get you out of using the ESC key sequence". If you are a poor typist and are continually mistyping CATALOG titles, you probably use the ESC IJKM keys to run up the Catalog and then along the row with the -> key. How many times do you forget to press R for the RUN twice or use some other means of getting out of the ESC sequence only to find you have typed UNMYPROGRAM and had SYNTAX ERROR come back at you. Well if you get into the habit of using ESC D as the last ESC character to move the cursor onto the line you want, there should be no such trouble.

Whilst on such facilities, do you know how to slow down or temporarily stop listings. A slow listing can be achieved by typing SPEED = 180 or some other number less than the normal 255. When you then LIST the letters will come out slowly. The lower the number the slower; with SPEED = 1 they take a very long time. Remember to reset to SPEED = 255 before you run the program.

To slow the listing down so that a few lines come up at a time before stopping, use CTRL S. Type LIST and press RETURN as normal, then put one finger on the CTRL key and keep it there. Then press the S key. Each time you press it the listing will either move or stop. In order to get back into BASIC type CTRL C. THIS WILL NOT WORK ON AN APPLE WITHOUT THE AUTOSTART ROM (including the IIT 2020) or if you are listing an INTEGER program.

Courses

Following the review of one of their courses in the last issue, we have heard from the University of Salford. They run three Apple II courses:

The Apple for beginners.
Getting more from your Apple.
Machine code programming on the Apple.

Their next courses run in January, April and July 1985 and during 1985 they are prepared to offer a discount of £10 per course to BASUG members.

Two books based on the courses are due for publication soon. They are "Getting the most from your Apple II/IIe/IIc" (Addison-Wesley) and "Machine-level programming on the Apple II/IIe" (Prentice/Hall). There is a disk associated with each book.

For further information please contact:

Conference Office
Maxwell Building
University of Salford
Salford
M5 4WT

Telephone: 061-736 5843 extension 449.

Assembly Course.

BOOK NOW

for

APPLE ASSEMBLY LANGUAGE COURSE

October 20th

County Hall, Central London.

A few places still left.

See separate application form.

BASIC programmers do it but get it wrong.

FORTH programmers do it in reverse (polish).

BBC Micro users do it down a tube.

Sinclair users can't do it without peripherals.

Sinclair users can't do it - full stop!

STOCK CLEARANCE

In order to make room for new stock the following items are offered for sale at lowest ever prices. Everything must be sold so any offer made will be considered.

N.B. VAT INCLUDED IN ALL PRICES!

USED HARDWARE

APPLE II EUROPLUS £275
APPLE DISC DRIVES FROM £145
CONTROLLER CARD £40
PARALLEL PRINTER CARD £35
SERIAL PRINTER CARD £30
COMMS CARD £30
DIPLOMAT COMMS CARD £50
SMARTERM 80-COL £75
DIGITEK 80-COL £75
DOS 3.2 CONTROLLER CARD £20
PAL CARD £25
ITT 2020 £195
ITT DISC DRIVE (APPLE COMPATIBLE)
£125
MONITORS FROM £35
VARIOUS PRINTERS FROM £145
APPLE POWER SUPPLIES £50
110V POWER SUPPLIES £30
GAMES SOCKET EXTENDER £4
ORBIT ACCOUNTS PACKAGE £200
APPLE II CARRYING CASE £35
APPLE II CARRYING BAG £5

COMPLETE APPLE II SPARES KIT INCL.
MOTHERBOARD, KEYBOARD,
POWER SUPPLY, DISC DRIVE, ALL
ROMS, RAMS & TTL CHIPS, SERVICE
MANUAL ETC. £750 OR WILL SPLIT
Ile PADDLES £15
APPLE II PADDLES £5
Ile 80-COL CARD £50
APPLE II AUTO-REPEAT KEYBOARD
MOD £10
MODULATOR £10
APPLEBAG £7.50
DUST COVERS FROM £4.50
DESKTOP WORK STATION WITH 4
SWITCHED 13A SOCKETS £20
NEW 13A PLUGS £7.95 for 20
SILENTYPE PRINTER £30
APPLE II MANUALS £3 EACH
VERSAWRITER GRAPHICS SYSTEM
£50
KOALA TOUCH-PAD FOR Ile £45
ALF MUSIC SYNTHESISER £95

USED SOFTWARE (ALL ORIGINAL)

ADVANCED VISICALC £100
VISICALC £50
DESKTOP PLAN £15
FORTRAN £40
WORDSTAR £90

MICROSOFT CP/M AND MANUALS (NO
CARD) £25
WORD HANDLER II £25
STAR TEXT (40/80 COL) £50
MAGIC WINDOW (40 COL) £45

MANY GAMES HALF PRICE - 'PHONE FOR DETAILS

SPECIAL OFFER FLOPPY DISCS

BULK PACKAGED (NO SLEEVES OR LABELS) DOUBLE SIDED/
DOUBLE DENSITY 5.25" DISKETTES. 5 YEAR GUARANTEE.

100 OR MORE	£1.00 EACH
10-99	£1.25 EACH

SLEEVES AVAILABLE AT 10p/DISC EXTRA

ADD £1.80 P & P

ALL PRICES INCLUSIVE OF VAT

BARCLAYCARD

ACCESS

BERKSHIRE MICROS LTD - BRACKNELL (0344) 484423

Book Reviews

Title: Applesoft Language

Authors: Blackwood B. D. & Blackwood G. H.

Publisher: Howard W. Sams

Price: £11.85

Paperback, spiral bound, 9" x 6", 274 pp.

First publ. 1981, 2nd edn. 1983

Reviewed by Brian Whalley

The first edition of this book was briefly reviewed for Hardcore in December 1981, but it is difficult to know just how the second edition differs from the first since there is no comment in either the Preface or the Introduction, and no special foreword to this edition. The original review commented on the lack of Index, and this has been remedied. Surely this alone isn't the reason for the price going up by just over 50% (from the original £7.65) in only two years?

George Blackwood claims in his introduction that the book is for beginners, for those who "want to learn how to program and have no opportunity for formal instruction", and who have "very little comprehension of computer programming". The chapters are termed "Lessons": Lesson 1 teaches the reader to load and save programs to tape, whilst Lesson 2 covers booting the disk operating system, loading and saving programs, and other DOS operations (such as CATALOG, RENAME, LOCK). Each lesson starts with a useful "Vocabulary" of the terms to be covered, but a genuine beginner who needs to learn the fundamentals may be feeling somewhat confused even as soon as the second page of Lesson 1, from reading the first and last of the twelve definitions for this opening chapter:

"CRT - This abbreviation stands for cathode ray tube. The picture tube in your television is a CRT. The CRT can be used to display the output from the typed-in input to your Apple computer".

"VDM - This abbreviation stands for video display module, which is an electronic screen for displaying data or information".

The second definition in the Vocabulary section for Lesson 2 would rapidly produce indigestion in any but the most enthusiastic of novice readers:

"DISK or DISKETTE - A magnetic disk is a storage device that consists of a flat circular plate coated on both sides with

some material (Mylar) that can be magnetized. The Apple II uses a single density, soft sector, 5 1/4 inch diskette as its virtual storage medium. The ...disk has a storage capacity of 118,000 bytes in the 3.2 disk operating system, and 146,000 bytes in the 3.3 disk operating system. The DOS 3.3 will store between 100 and 120 pages of normal text...."

... and it continues with figures for tracks and sectors and allocations for data and the directory. The Vocabulary does not define byte, sector or track, nor do those words appear in the book's index.

The Blackwoods do appear to subscribe to tidy screen displays, with details of HOME, HTAB, VTAB and TAB as early as Lesson 4. Lessons 5 and 6 hurry the reader along to descriptions and discussions of variables and the precedence of arithmetic operators respectively. Lesson 8 plunges the reader into Relational and Logical Operators, whilst Lesson 9 introduces Problem Solving and Flowcharts - which seems a curious point at which to introduce a definition of "hardware":

"HARDWARE - This is the name for all the physical units of a computer system. Hardware is made up of the apparatus rather than the programs".

Lessons 12 and 13 cover single and double subscripted variables whilst Lesson 14 on String Arrays helps us to print on the screen various bits of the memorable sentence "HI SUE" before we settle down to the detailed flowcharting and code writing to solve a real problem: "to input three lines of variable length separated by a delimiter (;) to allow any length of name, any length of street number and address, and any length of city, state and zip code up to 255 characters each". Is that immediately clear? If so, then perhaps a misprint has crept into this review: What, then, are we to make of this curious book? To be fair to the authors, there's a good deal of common sense in it, and clear explanations of some of the complexities of programming. To be fair to potential readers, however, the good material needs careful seeking, and there's a great deal to confuse and dismay the newcomer to programming. The semi-experienced microcomputer owner could be helped forward provided that he or she is blessed with patience, and the ability to tolerate some very curious bits of English that are more Blackwoodian than American, but this is NOT a beginner's book, unless

that beginner is very serious indeed and is deprived of access to the many better introductory books available to the Apple user.

Title: Apple BASIC Made Easy
Authors: Gardner D. A. & Gardner M. L.
Publisher: Prentice-Hall
Price: £14.35
Paperback, 9" x 7", 246 pp.

Reviewed by Brian Whalley

This is an attractively produced book for the beginner in BASIC programming. David and Marianne Gardner are justified in their claim that the book "will teach you how to write BASIC programs on the Apple II or Apple II Plus" and that there is no need to know anything about computers or to have any mathematical skill. The newcomer with an Apple, the Apple manuals, and the will to follow the book carefully, can certainly achieve the two goals that the authors set down of learning to program well and having fun doing it. There are some good diagrams, and a sprinkling of cartoons which are likely to appeal to the excited younger programmer even if not so much to the anxious adult!

The book starts fairly conventionally with an introduction to the keyboard, PRINT statements, and the use of NEW, RUN and LIST. Chapter 4 has a quick look at graphics, whilst Chapter 5 returns the reader briefly to hardware matters: the use of the disk drive or the cassette recorder. The topics which follow are found in any BASIC programming book: variables, FOR... NEXT loops, IF statements, sound effects and subroutines, GOTO, strings and string variables, and the rest. There are, however some interesting features. Each chapter, for instance, is carefully structured, starting with an introduction to what it covers and ending with a clear summary of what has been learned, to which is appended a set of exercises. The exercises are nicely devised to provide interest for programmers of pretty well any age or background, and offer genuine problems without posing frustrating difficulties. Many of them could be of considerable value in any course in BASIC programming, whatever make of system is used, and are much more stimulating than some other so-called exercises I've seen perpetrated on beginners.

Another interesting feature is a description of what the Gardners call "pseudocode", which is in effect a form of flowcharting but without its air of threatening

technicality. Examples of simple program outlines are presented, along with a full program listing and a diagram to link the two, for example:

LINE NO. PSEUDOCODE

5-30	Set the graphics screen and color
35	For repetition = 1 to 10
40-90	Bounce ball from left to right
100-160	Bounce ball from right to left
165	Next repetition
170-190	Reset text and clear screen
200	End

Smaller parts of programs are expanded in pseudocode:

```
For DOT = 39 to 0 step -1
    Plot a green dot in position DOT,0
    Pause
    Erase the green dot at DOT,0
Next DOT
```

It's not a new idea, but it's well used by the Gardners, and not overdone.

Sound effects and hi-res graphics are well covered without getting too elaborate. There is a useful chapter on PEEKS, POKES and CALLS - the chapter summary consists of a list of those specifically covered, but I would like to have seen that supported by a fuller list, perhaps as an Appendix.

The book does in fact have five Appendices. The first covers disk initialisation and program transfer, and the newcomer to disk drives will find it useful in this easily located position. Appendix B is a shape-table generator which certainly works: it includes routines for disk and cassette storage and recovery of shapes, though I'm at a loss to understand why the routine to DISPLAY a shape after BLOADING or SHLOADING requires a back-track to Chapter 26. The third appendix consists of a clear description of the order in which the Apple carries out arithmetic processes, and an explanation of scientific notation which is clear if you are sufficiently motivated to concentrate hard. The final two appendices are a list of Reserved Words, and a Summary of Commands which could have been improved by including page references. There is a comprehensive if slightly patchy Index.

This is a useful addition to the books on Applesoft programming. Some of the sample programs are rather unimaginative, and some could work rather better, but that's true of some quite expensive software! I would like to have seen rather more attention given to

good screen layouts, but that also is true of many professional programs. For example, it's disappointing and/or frustrating to find that the oft-used demonstration of a Bouncing Ball performs merely across the top line of the screen; to find line 120 listed as "FOR PAUSE = 1 TO 25: REM NEXT PAUSE" when "NEXT PAUSE" should be part of the program (as in a correct partial listing six pages earlier); to find curious line number sequences such as 160, 162, 163, 170; and to find the program using GOSUB 900 when Line 900 is a REM statement.

I continue to be puzzled at the variations in price amongst paperback microcomputer books of similar size and quality of production: this one is surprisingly expensive. However, it is a book that most beginners of any age would find both instructive and interesting, and one from which teachers of programming could garner some useful approaches.

Title: The Apple House
Author: John Blankenship
Price: £13.95
Publisher: Prentice/Hall
Paperback : 6.5" x 9" : 153 pages

Reviewed by Jason W. Smith

If you ever dreamed of having your house controlled by your Apple, then this is the book to buy! It details how to use your Apple in conjunction with various pieces of hardware to control your home for you.

It makes very interesting reading for both the technically-minded and the not so technically-minded Apple owner, as it opens up some very exciting possibilities for you and your Apple. After all, imagine talking to your house and having it respond by turning on or off the stereo or television and such-like things. Wouldn't it be fun to be able to have lights turned on as you enter a room or turned off when nobody is around? Maybe you have often thought it would be nice to have a phone-answering machine but didn't buy one because of the financial outlay. It could be accomplished using your Apple and two old tape recorders. What about a real-time clock, burglar alarm, speech synthesis and voice recognition. John Blankenship covers all these topics in great detail throughout this book to give a complete and workable intelligent home-control system which can all be controlled by your trusty Apple II.

All these facilities certainly sound very exciting, but there always has to be some

minor requirement that you must fulfil, and the requirement for this book is that you should at least have some general knowledge of electronics as a minimum. Many of the pieces of equipment needed can be purchased from various sources (a list of vendors is given at the back of the book), but if you happen to like construction projects, and aren't too bad with a soldering iron, you can save money by building some sections yourself. Complete schematics are given for some of the more appropriate areas. There are a few bits and pieces in the book that cannot be purchased anywhere and if you want them, you will have to make them yourself. The author recommends a number of appropriate books on the subject of hobby electronics and the Apple II.

The home control system described in the book has five basic functions. These are: Voice request, Phone control, Security management, Event timing and the monitoring of movement throughout the house. The VOICE REQUEST module allows communication with the computer using a wireless microphone. With this module you can control the lighting, ask what time it is, turn on the security system, turn appliances on and off and much more. The PHONE CONTROL module will turn the Apple into an intelligent answering machine. It also allows you to contact the computer from a distant location by phone and request information on the status of your house when you're not there or instruct the computer to perform some operation such as turning lights on or off. The module for SECURITY MANAGEMENT monitors doors and windows and, because the system is intelligent, you could have it phone a neighbour if it detected an intruder. EVENT TIMING provides a means of establishing a list of events that need to occur on an ongoing basis. For example, the computer might wake you at seven o'clock on weekdays but let you sleep until ten at weekends. The heating should be turned down or off when no-one is home and then back on just before someone arrives home. This information is held in tables which can be edited depending on how your lifestyle changes. The INTERNAL MOVEMENT module watches various sensors to keep track of where people are in the house. This information is used to decide what lights should be turned on or off as people enter or leave rooms. John Blankenship states that it is possible to use these modules alone, all together or with the addition of some of your own. This is due to the flexibility of the software provided, which is written in Applesoft.

The book is divided into four parts. Part 1

NEW DERING 10MByte HARD DISK DRIVE FOR APPLE COMPUTERS - only £1095



mitel
hard disk

- 10 MBytes formatted capacity
- 4 times faster than floppies
- Full back-up CP/M software included
- Runs with DOS 3.3 and CP/M
- Error-free data transfer
- Up to 5 MByte size CP/M files
- Contract maintenance available



The new DERING HD1 hard drive offers a massive 10 MByte storage capacity for all Apple and Apple compatible computers at an unbeatable price. It comes complete with Keele Codes CLIP back-up and compress software for CP/M.

Disk space is user definable between DOS 3.3 and CP/M giving a maximum size of 5 MByte for a CP/M file. It is supplied complete with interface card, controller, power supply, cased ready to go. It is the work horse of any system.

DERING HARD DISK
Unit 221
Dewsbury, Wetherby
Phone (0924) 499366, ext. EE
Telex 83147 VIA OR, Attn. DERING

is an introduction and an overview of the entire system. Part II describes all the hardware that will be required to make the system operational. Pieces of hardware that cannot be purchased are examined in greater detail than off-the-shelf items. Part III pertains to the actual program coding used to implement each program module. I must say that the main BASIC program is well structured and easy to follow, but I do not like the idea of program lines ascending in increments of one, two and sometimes five as this could be rather tiresome to type in. Luckily you can purchase an un-protected disk with all the software in the book for \$12.50. Part IV shows you how to put the hardware and software together to create your Apple-controlled house.

In summation I must say that I did find the book clearly written with no obvious errors in the text or in the programs supplied. This book has inspired me towards the idea of having the Apple controlling various things in my home. It is not a very large book at only 153 pages, but it certainly contains a great deal of information on turning your house into "the first Apple house on the block".

Title: Using the UCSD p-System
 Authors: K. Buckner, M. J. Cookson,
 A. I. Hinxman, A. Tate.
 Publisher: Addison-Wesley Publishing Co.
 Price: £11.95

Reviewed by Adam Broun.

This is a book aimed at "guiding the complete beginner through aspects of the (UCSD) system necessary for it to be used safely and sensibly". It deals with the UCSD system as a whole, examining the filer, editor, and other parts of the system, without being too specific about either language or machine. Of course, when one sees UCSD written somewhere, mentions of Pascal are never far behind, and the second part of the book relates to Pascal, insofar as it applies to the UCSD system.

My biggest worry about the book, which stayed with me the whole time I was reading it, was that I could never be sure at whom it was aimed. The first chapter is entitled "Basic concepts of computing" and goes on to provide a fairly standard, if rather convoluted introduction to the concepts of information storage, memory, etc., including explanations of how to put disks into drives. Subsequent chapters go on to using the system in great detail, and throw jargon about like it was going out of style. For

example, in chapter 4, which our computer novice has been told to turn to in order to adapt the system to his computer, he is casually told of "A miscellaneous information file (SYSTEM.MISCINFO) containing the terminal character sequences needed to perform various output operations ... and the character sequences to be interpreted on input as requesting some operation".

The book continues by giving thorough, if at times confusing, expositions of the rest of the system, and includes a few real pearls of wisdom, such as hints on how to undelete files. However most of the information covered is also in the standard Pascal manuals, and usually the manuals are a good deal easier to read, especially since they are Apple-specific. The book attempts to cover UCSD implementations for several machines.

To sum up: the book tries to do something very difficult, and probably fails - that is, to get a complete novice to computing to start off on the UCSD p-system. It may, however, be useful to the more experienced user in providing a condensed summary of the system.

Finally, a warning to those who still have the old version 2.1 system: Many of the features described in the book are not available on the older systems, and a brief summary of these is given in an appendix.

Keyboard Lowercase Mod.

Apple II+ keyboard - Lowercase modification.

by Dick Menhinick.

Some months ago, I reviewed a book called 'The Apple II Circuit Description' by Winston Gayler in Hardcore. In this review I mentioned a section in the book which described how Apple had allowed for upper and lowercase operation in the Rev 7 (two part) keyboard encoder but had failed to tell anyone about it!

Since then, a number of BASUG members have approached me and asked for details on how the Apple II+ keyboard can be modified to make use of this feature so I decided to modify my own keyboard and write the process up for Hardcore.

The KR3600 chip which Apple use in the

keyboard encoder has 9 outputs, of which only 7 are used. Bits 5 and 6 are normally the two most significant bits of the keyboard's output ASCII code. However, this chip has been preprogrammed with two sets of codes, and if bits 8 and 9 are substituted for bits 5 and 6 respectively, the other key code set may be accessed.

If you remove and examine your keyboard encoder board and compare it to figure 1, you should be able to identify the two 'bow-ties' or 'butterfly-pads' adjacent to the RESET/CTRL-RESET slide switch. These sit above and below six holes which are laid out to accept another slide switch of the same type as the RESET switch.

Examination of the circuit board, shows that the two 'butterfly-pads' connect the data bits 5 and 6 with the outputs 5 and 6 on the encoder chip. However, if the other two contacts (the ones not connected to the 'butterfly-pads') are examined, you will see that these are connected with the encoder chip outputs 8 and 9.

Now, if you carefully cut the copper track between the two 'arrow-heads' of both butterfly pads and fit a suitable double-pole double-throw (d.p.d.t) switch you will have the facility to switch between the standard Apple 'uppercase-only' keyboard, and a full upper and lowercase keyboard where the 'shift' keys work correctly! figure 2.

I fitted my switch off the board using long connecting wires, so that I could mount it through the metal baseplate of my Apple just under the keyboard. This removed the necessity to open the lid every time I needed to change the position of the switch.

Software-Support. Of course, as soon as I had modified my keyboard I wanted to try it, so, after carefully reassembling the machine I switched on, loaded DOS and tried to type some BASIC lines in lowercase. It didn't work!

"Rats! I must have the switch in the wrong position" I thought. I switched to the other position. I tried a couple of PRINT statements - it didn't work!

"***%+!@** Apple!!!" I expleted!

One of my greatest strengths is my unwavering ability to recognise when I am beaten. So I gave up and did something else.

A few days later, I decided to have a look at the latest messages on the Bulletin Board. Out came my trusty 'ASCII Express' disk and I booted up, dialed 0742 667983 (as it was) and waited.....! Up came the header and the greeting, and the acknowledgements, and the waffle and ... and ... and ... at last came the invitation to log in. I typed in my name.

IT WAS IN LOWERCASE ON THE SCREEN!

The dreadful thought then struck...."You Berk!" I mused.

In case you are now wondering what the hell I am going on about, I should point out that the good-old-trusty Apple Monitor keyboard input routine converts all lowercase keyboard codes into uppercase! Of course this routine is used by Applesoft which explained my earlier frustration.

The solution seemed too easy! So you will find, that unless you are using a case-sensitive screen editor like P.L.E you cannot put lowercase prompts into your programs! However, if you have a GET statement or are PEEKing the keyboard register you will get the lowercase you richly deserve.

The following programs will accept lowercase from the keyboard. This list only mentions the programs that I have personally tried - there are undoubtedly many more.

Mousepaint, ASCII Express, P.L.E., G.P.L.E., Magicalc (The Spreadsheet), Micro-Painter, P.F.S.file, P.F.S.reports.

The two illustrations were drawn using 'Mousepaint' on an Apple II+.

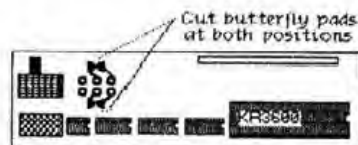


figure 1

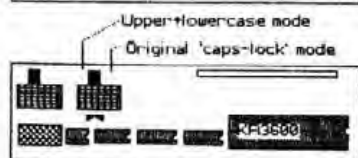


figure 2

Children's Software

LETTERS AND EARLY READING.

by G.G.Wood.

The recent challenge by Norah Arnold cost me many late evenings. Her request for children's programs encouraged me to complete a program for letter awareness and early spelling.

I started nine months ago with the Big Letters program by Ian Trackman which came on my Basug Introductory Disk. It was easy to add a patch so that any key pressing was echoed on the screen as a large letter. My three year old daughter was very happy pressing keys and recognising the letters: "M for mummy, D for daddy", she would shriek after finding them on the keyboard. The initials of her name all gave great excitement and real learning. She rapidly became fluent in all the letters, and spent many happy sessions on my knee gaining confidence and reinforcement of her Alphabet Books. She still loves to type random letter groups and ask me to say them.

"BUT", said my wife, "what about lower case letters?". "What indeed", I answered, as Ian's program is only upper case... Well, the first modification had been easy and Ian used a good modular layout with plenty of comment, so fond parent decided to further child's education.

First figure out the method of encoding letters (it's not x,y as a mathematician would assume), then develop a keyboard routine which emulates the case switching method of apple writer, then realise I need descenders so extend the letter array from 5*7 to 5*8, then selective erase to stop descenders interfering with the line below. Ian really knew what he was doing to implement only upper case!!

This took many long evenings and somewhere on the way I got fed up with the 30 second wait while the program loaded its arrays every time it was RUN. This turned into a real challenge, I finally solved it using the keyboard value as a pointer in the expression:-

ON key GOSUB n1,n2,n3,

with 45 different n's switching to routines for 45 of the letters.

The next evolution was to type words in upper and lower case linked into the original program feature for storing, editing and displaying complete messages (up to 99 lines). This allowed me to build stories based on words from the child's early reader. All the adventures of Dick and Dora are now part of her "big letters" program. My daughter loves reading them and the extra attention from other family members as she reads the words.

I can highly recommend this program to help pre-schoolers develop letter awareness and reinforce their early reading skills. Perhaps Basug will put it on the introduction disk or a childrens collection. I pass on the next challenge for another fond parent to add a simple picture function to enhance the stories.

Subject: Magic Spells

Publisher: The Learning Company
4370 Alpine Road
Portola Valley
CA 94025, USA

Price: £22.95 + VAT

Available from P & P Micro Distributors Ltd.

Reviewed by Elizabeth Raikes

This game is for 6 to 10 year olds. It tries to teach you to spell by unscrambling anagrams. If you get it right you win the gold. If you get it wrong the wizard of spells gives it to the demon. If you get all the spells right you win the treasure at the end of the rainbow. You can try the wizard's spells or make up your own lists of words.

Magic Spells is a fun game especially for me because I like it if the demon doesn't get any gold. I think it is good for children younger than six if you write lists of words which are very simple. Parents usually have to write the lists to make sure they are spelt correctly in the first place. I think it's worth playing and I play it quite a lot.

Admin.

THE ADMINISTRATOR MOVES HOUSE.

By now many of you will have experienced the tortuous route you have to travel via the telephone network in order to talk to me, or a slight delay in the processing of your orders. It is quite probable that you have also become very bad tempered in the process (I don't blame you) and wondered what the hell it was all about! Well, if you are sitting comfortably, let us begin.

Last year my fiance and myself decided to sell our respective houses and purchase one together. My house in St Albans went first and so BASUG moved to Bracknell. A few weeks ago we found ourselves in the unenviable position of having also sold the Bracknell house before completion on our new property. Oh, what panic set in. Not only did the furniture have to go into store, the dogs into kennels and a temporary home be found for us but what was going to happen to the BASUG office temporarily? After many hours of brow beating and tearing out of hair we found the solution. Bob Mould (a gallant member of the club) allowed us to have our telephone no. re-directed to his own line, enabling us to use an answering machine to

take messages; also the post was re-directed to his address. This meant that I could call in once a day, pick up the post and messages and deal with it in my fiance's office (who was very tolerant of the situation). But in the process of dismantling my office, setting it up temporarily and once again moving it into the new house I am afraid there have been short delays in answering the post for which I apologise.

As I write this I am waiting for the phone to ring to confirm that contracts have been exchanged on our new house and that completion will be on 14th September, then I shall jump on the Post Office and British Telecom and plead with them to change all the re-direction orders from Bob Mould's house to our own and with a bit of luck BASUG will be finally and permanently in business in a settled address with no more confusion. My new telephone no. is 0635 46385. Please cross out of your little black books ALL previous telephone nos., etc.

One final note, our packers were so highly efficient at swooping everything into their arms and into boxes that, in a couple of weeks time, I shall have to unpack 4 cooked sausages still in the frying pan that we were going to have for our lunch a few weeks ago! Oh, the joys of moving!!

APPLE GAMES EXCHANGE

ARE YOU

BORED with the same old Games

DO YOU

FIND the New Ones are OVERPRICED

NOW

YOU can exchange your OLD Arcade and Adventure Games from as little as £3.50

Send S.A.E. for full details together with a list of the games you wish to exchange to:-

APPLE GAMES EXCHANGE

P.O. BOX 21

DISS NORFOLK IP22 3DQ

BASIC Note

Education

Here is a short BASIC program which you may find interesting. Try typing it in and see what it does. If you write games you may find it valuable to study this technique. Thank you to Adam Broun who sent it in.

```
0 REM WRITTEN BY A.M.BROUN
10 HCOLOR= 7
20 GOTO 1000
30 HS = PEEK (230)
40 IF HS = 32 THEN 130
50 POKE - 16299,0
60 HS = 32
70 GOSUB 230
80 POKE 60,0: POKE 61,64: POKE 62,
  255: POKE 63,95: POKE 66,255: POKE
  67,31
90 CALL -468
100 HCOLOR= 0: HPLLOT 0,0 TO 279,0:
  HPLLOT 0,64 TO 30,64: HCOLOR= 3
110 POKE 24575,0
120 GOTO 210
130 POKE -16300,0
140 HCOLOR= 0:HPLLOT 0,0 TO 279,0:
  HPLLOT 0,64 TO 30,64: HCOLOR= 3
150 HS = 64
160 GOSUB 230
170 POKE 60,0: POKE 61,32: POKE 62,
  255: POKE 63,63: POKE 66,255: POKE
  67,63
180 CALL - 468
190 HCOLOR= 0: HPLLOT 0,0 TO 279,0:
  HCOLOR= 7
200 POKE 16383,0
210 POKE 230,HS
220 RETURN
230 CALL - 182
240 POKE 71,0
250 CALL - 193
260 RETURN
1000 Y = 80
1010 HGR2 : POKE - 16302,0
1020 FOR I = 0 TO 279
1030 Y = Y + RND (1) * 6 - 3: HPLLOT
  I,191 TO I,Y
1040 NEXT
1050 GOSUB 30
1060 FOR I = 272 TO 279
1070 IF Y + RND (1) * 6 - 3 > 279
  OR Y + RND (1) * 6 - 3 < 0 THEN Y
  = Y + 10
1080 Y = Y + RND (0) * 6 - 3: HPLLOT
  I,191 TO I,Y: HCOLOR= 0: HPLLOT I,0
  TO I,Y - 1: HCOLOR= 3
1090 NEXT
1100 GOTO 1050
```

by Norah Arnold

A Review

PROGRAM: CLASS MARKS
AUTHOR: ANTHONY GAME
PRICE: £20.00
CONTACT: Tel. 0394 282820

Every teacher, regardless of which age group they teach, spends a great deal of time writing out and manipulating lists of names and marks. The author of this program, being a retired teacher himself, has tried to produce something which will cut down the time spent over mark lists.

When the disk is first booted a title page is displayed and you are requested to press any key. This brings up the main menu which can be recalled at any time by pressing Control X. The first item on the main menu, Begin new list, can be accessed by pressing B. You are then asked to give the list a name and state how many pupils will be entered. Next you are prompted to enter the names and the marks. I found this section quick and easy to get through once I was used to obeying the conventions listed in the instructions, i.e. surname to be entered first, followed by one Christian name or an initial.

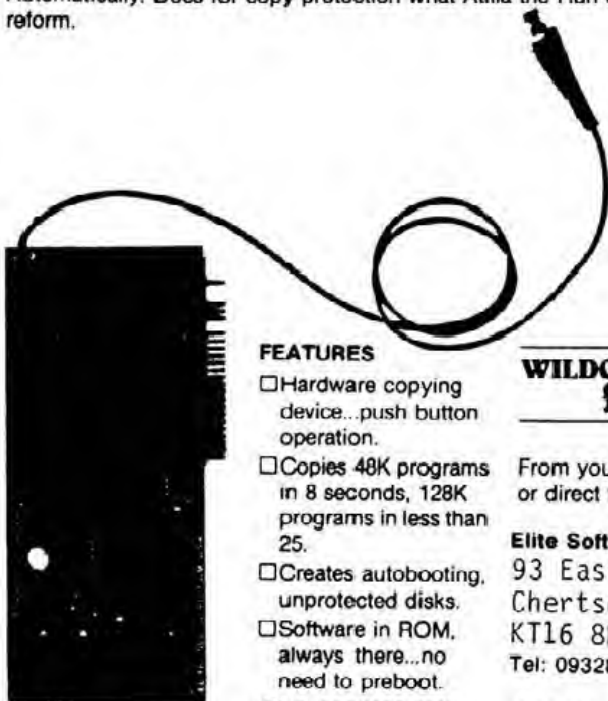
Although you are given an opportunity to check your typing immediately after entering the list, I found the next option on the menu -R Read list, handy for checking what I had actually got in the list. Other options on the menu allow you to add names to the list or delete names from the list. One can also change a mark on the list if it is found to have been entered wrongly.

The next item on the main menu is -X Change all marks. This item can save you a great deal of time if you work with the same pupils over several months. Having typed the names in once for one set of marks relating to one subject, the list can be recalled, renamed, and the marks for the new subject entered. Next come two items on the menu which help with sorting the marks and names. Pressing P puts the list of marks into numerical order and calculates positions, and it does it with reasonable speed. Pressing

WILDCARD PLUS

COPIES 64K PROGRAMS IN 12 SECONDS

The new **WILDCARD PLUS** is the card that thinks for itself. One push of the button, one simple menu - with its own 6502 processor **WILDCARD PLUS** makes a perfect copy on any Apple, from a 48K][to an extended 128K // e. Automatically. Does for copy protection what Atilla the Hun did for social reform.



FEATURES

- ☐ Hardware copying device...push button operation.
- ☐ Copies 48K programs in 8 seconds, 128K programs in less than 25.
- ☐ Creates autobooting, unprotected disks.
- ☐ Software in ROM, always there...no need to preboot.
- ☐ No language card required, works equally well with or without.
- ☐ Utilities include: compression software - several backups may be stored on a single disk, recover BASIC programs, disassemble machine code software, screen dumps, etc.

WILDCARD PLUS
£119

From your local dealer,
or direct from:

Elite Software Company
93 Eastworth Road
Chertsey, Surrey
KT16 8DX
Tel: 09328 67839

Add 15% VAT for
delivery in UK P & P
included.

Dealer enquiries
welcome.

Apple is a registered trademark of Apple Computers Inc.

O puts the names into alphabetical order. Both these options, to calculate positions and to put in alphabetical order, give you the chance to reverse Surnames and Christian names so that you could print Fred Bloggs, Bloggs Fred or even Bloggs F if you had entered an initial only. A point to watch is that the names must be Surname first when the list is saved to disk but this is explained in the instructions.

Next come two options which are very useful indeed. Main menu option M allows you to standardise marks by setting another mean and standard deviation, so enabling you to compare marks from different sources. Option T will let you present your marks as percentages. When using this option I found it best to save my original list to disk, recall it and change the name, then convert the marks to percentages, so that I still had the original marks if I needed them.

The disk menu gives the normal choices of catalog, saving a list or reading a list from disk. It also gives you the chance to save to a Totals list. This gives you the choice of keeping a marks list as a single set of marks, or of adding them to another list, called a Totals list, which adds the new marks to the previous ones and stores them as a total or an average. This may be done as often as necessary so that a whole term's totals or averages could be stored.

There are many good points about this program, not the least being the speed at which one can obtain a list of marks and names in order of position, always a tedious job if the list is long. The program is fairly easy to use at the beginning and very easy once you are familiar with it. One little useful point is that a list of names may be printed without the marks if necessary, giving a speedy way to produce a list for checking off which pupils have paid their outing money, for instance. I have always used Visi-Calc for keeping records of marks until now, but although Visi-Calc has a few advantages in that one can choose how many columns of marks or which part of a list to print, it is slow and tedious after using Class Marks. I shall still have to use Visi-Calc for some things, especially where a special look-up table is needed to interpret the marks, but for the everyday mark list I shall use Class Marks.

There are one or two little things that might be improved. It is very nice to have

the choice of upper or lower case for the names, but I met a difficulty with the lower case. I tried to put in the name O'Donnell and couldn't, neither could I put in McKivett. The apostrophe could not be entered in O'Donnell, and in both the names the second capital could not be entered correctly. I got round this by doing that list in upper case and leaving out the apostrophe in O'Donnell. Also it would be pleasant to have the choice of single or double spacing the lines of a printed list. These are small criticisms which I am sure the author will consider putting right, otherwise Class Marks is a very useful program for a teacher to have around.

EXAMPLE LIST

NAME	MARK	POS
Stephens Joan	89	1=
Inestyne Patrick	89	1=
Briggs John	82	3
Turner Catherine	79	4
Johnston Brian	70	5
Miller Carol	65	6
Lawson Peter	62	7
Hendrick Anne	57	8
Dysen Miles	53	9
Brandrick Audrey	52	10
Laurence Stephen	48	11=
Drench Joan	48	11=
Wright Joan	46	13
Marks Annette	45	14
Arnold James	44	15
Richardson Janet	41	16=
Jolly Charles	41	16=
Hawes Stanley	38	18
Bain Terence	36	19
Catterick Jean	33	20
Frankie Gillian	30	21
Mitchell William	29	22
Birch John	28	23
Arkwright Richard	25	24
Ackroyd Kenneth	24	25

Seedlings

Macintosh 512k Big Mac is promised by Apple for October 1st. It will cost £800 for an upgrade. This must mean that Lotus 1-2-3 is on the way.

Softalk have filed for bankruptcy.

Lister

Programming Utility

by Roger Harris

There are many sophisticated Apple utility programs which help to develop a new program or service an existing one; line editors, programs which renumber lines or find variable names or strings are examples. The following BASIC program, called Lister because it started out as a means of listing selected program lines with output to screen or printer, helps me to perform certain frequently used operations by using a RUN command followed by a one or two digit line number, e.g. RUN 1 instead of LIST 2500,2599. The line numbers, and their labelled keys, were chosen to minimise finger movement and to avoid clashing with key labels I've placed on keys 7, 8 and 9 which form part of a "numerical keypad" (see Windfall, April 1983, p. 22 for details and hex dump).

Lister occupies lines 1-12 and 60000-60200. Line 0 leads to the start of the program being developed or serviced which fits between lines 12 and 60000 and is thus embedded in Lister which may be deleted once work is complete. (Is it ever?!). A new program may simply be typed into Lister. An existing program may be merged with Lister by using a Merge Utility such as 'Renumber' (available on DOS 3.3 System Master disk) or APA (available on DOS Tool Kit disk).

The commands are as follows:

RUN 1: Lists on the screen a selected program segment. See 'RUN 11'.

RUN 2: Prints a program segment with a page heading showing the program name, date and time. The command asks for a REM to describe the segment and for a reprint/serial number (easier than retyping line 60150).

RUN 3: Type <RETURN> to set screen width to 33 columns. Type <3> to return to TEXT mode (40 columns).

RUN 4: Same as CALL-151 to enter MONITOR.

RUN 5: Same as CATALOG. Command asks for drive number, i.e. 1 or 2.

RUN 6: SAVES Lister and its embedded program to drive 1 or 2 using the name, date and time shown in lines 60100-60120, e.g. PROGRAMNAME.31AUG84.1635. Lister does not

check if the name length is more than the maximum of 30 characters.

RUN 11: Lists on the screen the lines showing the program name, date, time and also the line numbers of a program segment. The lines may be amended by re-typing using ESC IJKM to move the cursor.

RUN 12: Resets 'last drive used' to drive 1 or 2.

Since using Lister whilst programming each subroutine and segment now has its own page in a loose-leaf binder with its program name, REM, date and reprint number together with other descriptive data and notes. Updated versions of programs, when SAVED to disk, now have a consistent filename showing date and time to indicate "freshness".

```
0 GOTO < LINENUMBER > : REM SEE
      TEXT
1 GOSUB 60150: END : REM LIST
      SEGMENT
2 GOTO 60000: REM PRINTER
3 GOTO 60050: REM PAGE WIDTH
4 GOTO 60060: REM MONITOR
5 GOTO 60070: REM CATALOG
6 GOTO 60080: REM SAVE PROGRAM
11 GOTO 60090: REM LIST DETAILS
12 GOTO 60100: REM SELECT DRIVE
59999 END
60000 INPUT "REM ";R$: PRINT :
      INPUT "REP #";S$: GOSUB 60110
60010 PRINT CHR$(4);"PR#1";
      CHR$(13): POKE 1657,80
60020 PRINT "PROGRAM=";P$: SPC(9);
      "DATE=";D$:T$: SPC(9);
      "REPRINT=";S$: PRINT "REM=";R$
60030 GOSUB 60150
60040 PRINT CHR$(4);"PR#0": END
60050 GET Q$:D = VAL(Q$):
      POKE 33,33 + 7 * (D = 3): END
60060 CALL -151
60070 INPUT "CATALOG DRIVE #";D:
      PRINT CHR$(4);"CATALOG,D";
      (D = 1) + 2 * (D = 2): END
60080 GOSUB 60110: INVERSE : PRINT
      P$;D$;T$: NORMAL : INPUT "SAVE TO
      DRIVE #";D: PRINT CHR$(4);
      "SAVE";P$;D$;T$;",";D";
      (D = 1) + 2 * (D = 2): END
60090 LIST 60110,60130: LIST 60150:
      END
60100 INPUT "SELECT DRIVE #";D:
      POKE 43624,(1 + (D = 2)): END
60110 P$ = "LISTER"
60120 D$ = ".01SEP84"
60130 T$ = ".2200"
60140 RETURN
60150 LIST 00000,63000
60160 RETURN
```

DS:3

Reviewed by Bob Raikes.

Regular readers of Hardcore will have seen mention in the past of the group 'Mainframe', in particular an interview in the October '83 Hardcore. In that interview, mention was made of a 'drum machine' on the Apple being used on a record. The DS3 is the commercial product that has developed from this board. TV viewers may have seen the board on 'Me and my Micro' or 'Tomorrow's World'.

The DS3 consists of a board that fits into the Apple (Slot 5) and a cable which comes out of the Apple to a connection box to attach to your sound system. The full system includes a five octave full size musical keyboard, also on a cable from the Apple. There are 3 disks of software. To use the DS3, you need a 48K Apple with disk drives and paddles. The Apple can be of the II or IIe variety, if a II, then a 16K Language/RAM card is an asset. It will not work with an IIT2020.

The system works on the principle of digital recording or 'sound sampling'. In this process the fast changing voltage levels from a microphone, generated by an audio signal are converted into numbers. These numbers can be processed and stored and then converted back to an audio signal by a reverse process. The faster this is done, the more accurate the final signal is. A scientist called Nyquist established that to accurately reproduce any signal, it is necessary to sample it at twice the frequency of the highest signal to be recorded. In the case of sound up to the limit of most people's hearing at about 15KHz, this means sampling at 30KHz. The DS3 meets this criterion by sampling at that speed.

The recorded sound can be stored away on disk, and recalled at any time for translation into sound. One of the main problems involved in this system is the amount of memory used. For each second of sound, 30K of memory is used. This limits the maximum length of a sound recorded on the DS3 to between 1.5 and 2 seconds. If all you wished to do was record and replay a fixed sound, then this ability to record would be of limited value only. This is the reason that it was originally seen as a drum machine. Most drum kit sounds die away fairly rapidly so that it was possible to

fit in the Apple's memory three or four different drum sounds at the same time. A program to string them together was written and so a drum machine was made. Aspiring drummers (or their neighbours) will appreciate the advantage of a means of producing the sound of a drum kit electronically. The machine was first demonstrated to the BASUG Herts Local Group in more or less this form. Very impressive it was too.

However, having converted a sound into numbers, it is relatively simple to process the sound by editing it, or by changing its frequency. If you can change the frequency, you can reproduce the note at a different musical pitch, so the next step was to develop a keyboard interface to play the notes back. This was done, and software was written to make full use of it. What this means is that you can record ANY sound and play it back ... in four part harmony!!! The musical possibilities are limited only by your imagination.

Anyway, let's look at the facilities provided. From the main menu, the first option is 'Sound Sample'. This is the part of the program that records the sound. First a microphone has to be connected, via some kind of pre-amplifier (hi-fi amp, cassette deck, etc.) to the input socket of the DS3. This only accepts line level signals. The sound sampler, once activated, works by detecting any sound above a threshold level. This threshold can be set by the user. Alternatively the sampling can be started and stopped manually. When the sampler is not activated, the waveform of any sound being fed into the system is displayed on the high res screen as if on an oscilloscope. This allows the adjustment of sound levels to minimise distortion. Once the sample is complete, you can go into play mode and the sound can be played back from the Apple or musical keyboard. This is great fun, especially with children speaking or singing into the microphone. The final option in the sample section is 'trim'. This routine allows the beginning and ends of the sound to be trimmed both to save memory, and to avoid any time lags in the keyboard responding. To make trimming easy, the waveform is again shown on the high res screen.

The next facility from the main menu is waveform editing. As with the sample trimming, this shows a sample on the high res screen. This section allows the waveform to be changed to eliminate clicks at the start or end of the note, or to change the

GREENGATE

p r o d u c t i o n s



When you buy a Greengate DS:3 you are entering a new world. Not only are you providing yourself with a high quality 4-voice Sound Sampler to match the best around for all-in performance but you are into a new kind of sequencing which is **ONLY** possible with a full-blown computer system.

The Sampler speaks for itself ... But what Drum Machine or sequencer unit can:

- ☐ Give you **INSTANT** access to 8 individual 4-note polyphonic sequences while playing live?
- ☐ Give you **INSTANT** drop-ins (Or drop-outs)?
- ☐ Provide single or multi-sequence triggering?
- ☐ Allow synch to or from external source at all times?
- ☐ Provide you with **COMPLETE** records of your song sequences on floppy disc in digital, non-degradeable form? ... AND ...
- ☐ Remember the instrumentation details as well as the sequences?
- ☐ Let you substitute **ANY** instrument **ANYWHERE** in your sequence at any time—live or step...?

We don't know of one. But we **DO** know that the DS:3 is one of the very best drum sequencers. We have a collectors item 12" record to prove it ...YYY001.

Ask around for a copy ... or send us a PO for two pounds and it's on its way!

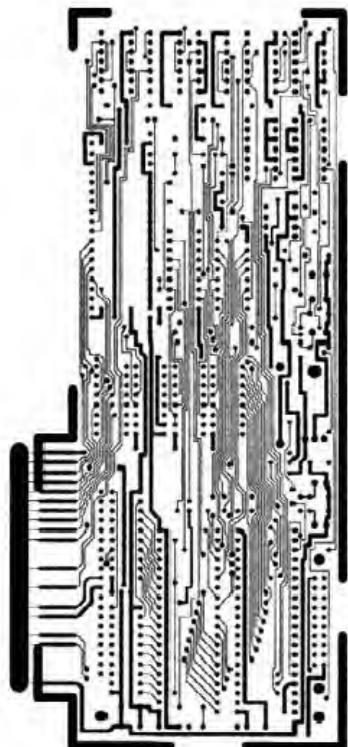
Sampling AND Sequencing? For £250?

The Greengate DS:3 does **JUST THAT...**

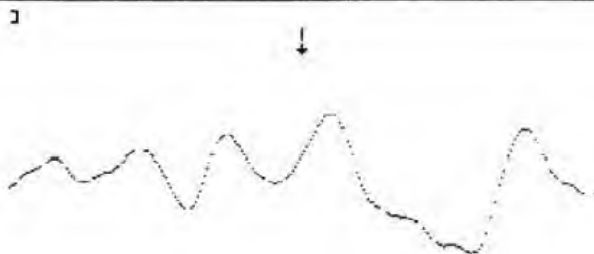
GREENGATE PRODUCTIONS and "MAINFRAME" announce their Digital Sound Sampling Sequencer, the DS:3. Starting at £250, this APPLE® based 4-Voice polyphonic system brings high-quality digital sampling within reach of every musician and studio.

15kHz bandwidth and two-second-plus sampling periods ensure excellent sound quality. Powerful sequencing software together with 5-octave keyboard control of pitch gives a percussion/drum kit capability limited only by the users imagination. There are no "PROMS", "Chips" or extras to buy!

DS:3



*APPLE is a trademark of Apple Computer Inc., Cupertino, CA, USA.



Sequence)
Sequence)
Sequence)
Sequence)--->Sequence + Sequence--->Sequence)
Sequence)

Sequence)
Sequence)
Sequence)
Sequence)

SETUP--
SOUNDS)

TEMPO)
etc.)

shape of a sample. Unfortunately, this can be a very slow process as each value has to be individually set. This section would look to me like a natural application for a Mouse to allow fast editing and perhaps creation of completely new waveforms.

The third option on the main menu is the Sound Play/Sequence option. This brings you to the other major part of the system. First you must select the sound(s) that are going to be used and the tempo. This part of the program allows you to 'record' the sequence of notes played against a metronome beat. The recorded sequence can then be played back or set to repeat. For example it is quite simple to record a 2-bar drum sequence and then set it to repeat. The resulting steady rhythm is very useful when practicing or 'composing' music. The sequencer has four different channels, so that four different sounds can be played simultaneously. For example, when recording a drum sequence, it is best to sound each type of drum or cymbal on a different channel so that each sounds for its full time.

Each sequence on each channel can be saved and allocated to a different key on the Apple keyboard. The sequences can then be merged so that they are played simultaneously and the merged group can be assigned to a new key. Up to nine different sequences can be available from the keyboard at any one time. Sequences can be appended to each other to form new longer sequences. Each of the sequences is saved to disk

PERFORMANCE -----< (song file <---
(song file
(song file

separately. There is a further 'chain' mode that will record the order in which sequences are played and allow this information to be saved and again allocated to a key or saved to disk.

A 'set-up' file can be saved, and this will detail the sounds used, the sequences and chains used and the tempo etc. A 'set-up' file can be converted into a 'song file', and these can be combined into a 'performance file'. By this point, you may be as confused as I was when I first tried to get to grips with the different types of files, and a diagram may be helpful.

The top of this tree file structure is the performance file. If this is used for live music production, it will play each song in sequence, automatically loading the new sounds, parameters and sequences, waiting for one key press between each song. The main limitation to this is disk capacity. Although sequences and other data do not take up a great deal of room, the sound samples do. This means that if you are creating performance set ups with many different sounds, you will very quickly fill disks. Song and Performance files are created using separate options from the main menu.

There is an option labelled 'Sequence Develop' on the main menu. This seems to be the beginning of a fairly powerful sequence editor. At the moment, you can merge, chain and list sequences to the screen or printer. You can also add or delete spaces between events. It is not possible to change or delete an event, nor enter sequences from the editor.

The final area available is the facility to change the allocation of notes on the Apple keyboard. This is important when using the non-keyboard version of the DS3, as the Apple keyboard can be used like a musical keyboard. New set-ups can be saved as files.

In use, the system is the first system I have seen, other than the Alpha Syntauri, that is usable for practical music making. It does all that it claims. The quality of reproduced sound is very good, but, and it is an important but, like any recording system, the sample is only as good as the skill of the person who recorded it. A good quality microphone, and some idea of microphone technique are both needed to achieve the highest quality results. Access is also needed to instruments as required. Good basic drum sounds and some instruments come with the system, and I understand that further disks of sounds are available from the manufacturers for those who do not have access to instruments.

The sounds are very realistic when reproduced at the pitch at which they were recorded, but many sounds then have only a limited range before becoming either unrecognisable or occasionally unpleasant sounds. Another feature that can affect the realism of the sound is the change in the length of the sound with the pitch. This means that sounds can become very short in the higher octaves. The present version of the software allows a sound only to be allocated to A (440Hz). This can be limiting if the pitch of the sound source is not the same.

I found the ability to synchronise to tape very useful. The drum kit I was using (Bass drum, Snare drum, Open Hi-Hat, Closed Hi-Hat, Crash cymbal) filled memory, so when I recorded this using a 4 track Portastudio, I recorded the synchronisation track on track 4, the drums on track 1. Each channel from the DS3 was fed to one input on the mixer where separate equalisation and effects could be applied to each. Then a

bass sound was sampled, a sequence to fit the song developed, and then this was recorded on track 2, synchronised to the drum track. Although I chose not to use the facility, I could then have recorded a rhythm sound on track 3, again in synch.

I did find the software irritating at times. It was clearly written as each facility was developed, so there was considerable inconsistency in the effect of different keys. Sometimes a return was needed after selecting an option, sometimes not. In most of the program, a control-c invoked a small menu that allowed the disk to be catalogued or the main menu to be accessed, but not always. Sometimes control-c was used for another purpose. When the control-c menu is accessed, the current drive is not shown, so frequently when using a 2 drive system the wrong drive was catalogued. More importantly, there is no facility to initialise a disk from within the program, so that a carefully recorded sound could be lost if a disk with enough space is not available. This is a consideration when a sound may be 130 sectors or more. A further problem is the tendency of the program to crash for no apparent reason. To be fair, so far I have not lost data but on one occasion I hit reset after crashing. The program appeared to restart, but the DOS had been disconnected, so data could not be saved.

The manual has been written by someone too close to the program, and who was using an Apple II+. I was using a //e, and found the keyboard instructions confusing. There is also no Index.

CONCLUSION

Once I got the hang of the system, I found it quite easy to use and very powerful. It really is a practical music making system. It would be useful to anyone involved in making and recording modern music. The hardware is to a very high standard, but the software is irritating although very powerful. The cost of the system is very reasonable, the basic unit being £250 + VAT (with the keyboard £450 + VAT). I am pleased to say that the cost includes 2 free software updates. Subsequent updates will be available at a reasonable charge.

(I am hoping to buy a system for my own use as a result of my experiences).

Sorting

QUICK AND SHORT SORTS

by R. C. Lowe

The most powerful sorting algorithm that I know of is the QuickSort. It is usually implemented as a recursive routine but here is a non recursive version that you may find useful.

It sorts the array NM(X) which has QN numbers in it.

```
1 QM = 12: DIM QL(QM),QR(QM):QS = 1:
  IQL(1) = 1:QR(1) = QN
2 QL = QL(QS):QR = QR(QS):
  QS = QS + 1
3 Q1 = QL:QJ = QR:QX =
  NM( INT ((QL + QR) / 2))
4 IF NM(Q1) < QX THEN Q1 = Q1 + 1:
  GOTO 4
5 IF QX < NM(QJ) THEN QJ = QJ - 1:
  GOTO 5
6 IF Q1 > QJ THEN 9
7 QWS = NM(Q1):NM(Q1) = NM(QJ):
  NM(QJ) = QWS
8 Q1 = Q1 + 1:QJ = QJ - 1
9 IF Q1 < = QJ THEN 4
10 IF Q1 > = QR THEN 12
11 QS = QS + 1:QL(QS) = Q1:
  QR(QS) = QR
12 QR = QJ: IF QL < QR THEN 3
13 IF QS < > 0 THEN 2
14 RETURN
```

For a short sort the best I have found is this version of the shell-metznar sort which fits on two lines. It uses NM(X) and QM in the same way.

```
1 QM = INT (QN / 2): FOR QA = - 1
  TO 0 STEP 0: FOR QJ = 1 TO QM - QM:
  FOR QH = QJ TO 0 STEP - QM:QL = QH
  + QM:QK = 0: IF NM(QH) > NM(QH)
  THEN QZ = NM(QH):NM(QH) = NM(QL):
  NM(QL) = QZ:QK = QH
2 QH = QK: NEXT: NEXT:QM = INT(QM / 2):
  QA = (QM > 0): NEXT
```

Is it possible to have a one line sort? If it is I am sure somebody out there has one so why not write in to Hardcore and tell everybody about it?

Readers' Letters

Guisborough, Cleveland.

Dear Sir,

I spoke to Nik Kelly about the BEEB/Apple software and he mentioned a 'game port' printer link but could not locate the article. I wonder if anyone could tell me who was involved with this and possibly send me the article.

Could you please let me know of any user groups in my area.

Are there any Wizardry fans, i.e. people who wouldn't mind being rung up or ringing up at 1.00 am for help/advice on 'Legacy of LLYYGAMYN' (any fans will know what I mean). I'm on ~~01652 881111~~.

Yours faithfully,

T. I. Young,

/Ed. -Dick Pilgrim is trying to set up a local group. Contact him at 20 Woodley Grove, Ormesby, Middlesbrough./

Sunbury-on-Thames, Middlesex.

Dear Yvette,

I notice that in Quentin Reidford's review of "Data Capture 4.0" in your February (84) issue, he expresses surprise that there was no facility for changing data word length...".

As the owner of an early (1980) copy of "DCAP 4.0/80", I happen to know this is not in fact true, though no-one could be blamed for missing the point - it's not in the documentation, which doesn't even tell you what the default values are, but tucked away in a "Help File" on my disk is a cryptic note headed "Changing Parity and Word Length" (that's one way to do it, I suppose!) This gives, for the Micromodem II, two addresses into which to poke "X", "the value found on page 37 of the Micromodem II manual"; and for the Communications Card, two addresses into which to poke "X", which in this case is not specified at all! I have the AIO Serial/Parallel Card, which appears to get the same treatment as the (Apple)

Communications Card, and it would seem that "X" is a value specified for the 6850 ACIA which both cards use, values for which are tabulated in the AIO manual.

It's all very sketchy and unsatisfactory, but as Quentin said, one doesn't usually have to make the change and I never have done. It may be that the "Help File" on later versions may be more explicit - I believe Southwestern Software do keep improving it. And it's always possible that some of our dealers know.

Yours sincerely,

R. Teale.

London NW1.

Re Andy Holderness's ITT 2020 article, Hardcore 4/3.

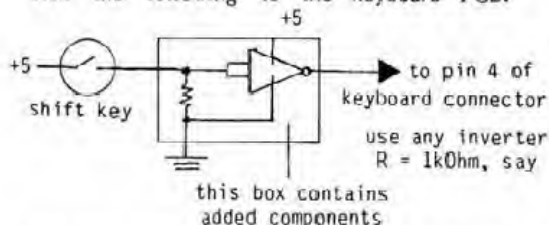
b) The Videx Softswitch works.

a) Lower case mod. can be done.

a1) Display: a 2716 EPROM on a piggyback board can be used to replace the character generator: we got ours from Lion House long ago.

a2) Shift Key. A shift key mod compatible with the 'standard' Apple II mod can be made. For 2020s with (newer) cherry keyboards:

Add the following to the keyboard PCB:



On main board, lift pin 1 of H14 (74LS251); connect wire from keyboard connector pin 4 to the lifted pin. (This will affect ITT graphics ninth-bit if graphics mod not done).

The chip added to the keyboard can be 'piggybacked' onto an existing chip via +5V and ground pins.

Older (Apple) keyboard: make connection to pin 4 of the keyboard connector as described in Apple literature.

Mike Salem (Hilderbay Ltd).

Swansea.

Dear Gentlemen,

I want to ask you about a condition setting of EPSON FX-80. I am using an Apple //c and an Epson FX-80 with intelligent serial interface, because Apple //c has only serial interfaces.

As far as running word processing software goes, for example "Appleworks" et al., my Epson prints out very well, but it is impossible using "Mousepaint" software, written by Apple. I think this software's printout program is written only for Apple Imagewriter since in the manual there is no explanation of installation for Epson FX-80. I asked Apple UK but they have no answer. I suppose they want to sell their own printer so they don't want to teach me. As you know the Applepaint software is quite popular. If it can't print out on Epson, everyone has to buy an Imagewriter.

So please can you tell me how to set the Epson FX-80 for Mousepaint. For Appleworks I use these conditions: 8 bit, disable parity, odd, 9600 baud, 166/1124.

I thank you in advance.

T. Murakami.

Sevenoaks, Kent.

Dear All,

Re: Macintosh.

You really must stop everyone talking about Apple and differentiate between types of Apple; Mac, 2e, 2c, etc. (Ed. Point taken).

Can we please get some technical/operating hints together for the Mac.

1) I have considerable trouble with the DOS; badly need a DOS listing for error reports; ID26, No Memory space for ejecting disk, F0?????, etc. when disk appears faulty. Also OS does NOT (!) ask/offer to Initialise Disk, just ejects and refuses to accept it if one has copied some progs onto it but forgot to initialise first! I have sent one such to P & P Micro Distributors, a disk which machine rejects so I can't erase it or anything.

2) Errors or parts of Mac manual which don't operate as described therein:-

p.40 - (4) Not true. One can call for alarm and OS will call for another disk which appears entirely unrelated (!) AND one can't escape or Break from Alarm option, i.e. forget I asked for it - have to switch off.

p.41 (4) I can't CUT date from CP into Note pad. Black area of say "DAY" in calendar clicked and on attempting to drag all the numbers of "DATE", won't have it, so can't cut and paste as stated. Then can't get rid of black area around DAY without closing whole CP window.

p.95 Again, the stated facility that one can paste the clock into a doc. doesn't work for me; I can't pick up the whole date/time to cut into anywhere, only 1 numeral at a time will go inverse.

The other worrying thing is disks renaming themselves before/after switchoff and on switching on tonight finding clock has lost 5 mins. I did not inadvertently touch keyboard as I poke it around the LHS of Mac to leave desk front clear.

Secondly, after almost two months of owning a Mac, I am short of: 2nd disk drive - no printer buffer on offer(?) - No language, except Forth which I really don't want to put in front of Basic - No MacMultifile - No MacMultiplan - so I can't get on with even a domestic database, let alone some business experimentation which is now getting urgent; demonstrations by me of Mac's abilities in this sphere are awaited.

Thirdly, what news of Macworks please? I believe it's in the offing, but nothing definite, like the others above.

Fourthly, your Intro Disk is fine and I see the new Rev of Systems disk incorporates the very disk copy routine you have on there; the copying before that was received was chronic, another reason for the exasperation over lack of disk drive 2.

Fifth, when may we expect to see a Universal modem for the Mac. I would like to get on to Prestel and the bulletin boards as soon as possible. Perhaps Vicom could be chivvied along a bit.

Sixth, Mac Alarm by Tom Warrick was interesting but I had sussed it out without too much trouble. Watchout if you are working in say a Macpaint window with Clickart, and you call up the alarm window. My Clickart screen froze and I could get an Edit menu only, till friend Vernon

Quaintance had a look, cancelled the alarm window and the Paint window came alive again! For more odd things about alarm clock see the current (as at mid-August) Byte, p.248 They also have an up-to-date review of Mac. The request for extraordinary disks by OS which appear entirely unrelated to what one is currently doing is partly explained in Byte.

A good book in my opinion is the one from Microsoft (UK have it) by Cary Lu called "The Apple Mac Book"; a superb tome and worth every penny of £19.95.

Yours sincerely,

P. Knight, C.Eng.

Dinas Powis, South Glamorgan.

Dear Sir,

I use an Apple II Europlus to keep medical records, accounts, and word processing, using two standard Apple disk drives, a Juki daisywheel printer and a Sup'r Terminal 80 column display. I now have two problems.

I use PFS as my database for keeping medical records and accounts. At first it was fine as it is an easy system to use, economical with disk space and quite powerful in its ability to search the file. I have found however that now I have several hundred records on each disk its sequential approach to searches in the absence of indices makes finding a record or a series of records a time consuming business. It requires a 40 column screen, a further disadvantage, and I lack flexibility in data management, such as the ability to perform a global update, keep more than one database on a given disk and write from the database to another set of records. I find that unless I go to CP/M there seems to be a dearth of really good Apple database software. Can you advise me? Your response will be affected by the second problem.

I have an awful lot of data now, mainly text in the form of medical records, which must be rapidly accessed when a patient comes in. Apple disk drives are so limited in their capacity that I am having to change disks all the time. I do not wish to buy a hard disk system, not this year anyhow, and wonder if there is any way of enabling Apple to talk to double sided double density disk drives? I suppose this would require a change of operating system to CP/M. Do you

have any views on this? Would Omnis, for example, using Pascal and therefore independent of DOS be able to address a larger floppy disk facility than the standard drive?

I have experience of dBase II in another application on a different machine and quite like it but clearly it has its limitations for this type of application and I am unsure whether or not Apple using it could address a larger disk format. I have looked at Omnis, which is fine for my application, but seems to be limited to Apple disk format, or requires a hard disk. I would be interested to know whether it is in use with another, larger type of floppy.

What I really would like is a system where my word processor could read the database, and vice versa, and both use the 80 col. screen, a fast and comprehensive database program and of course the ability to convert my present records to the new system without re-entering it by hand! No doubt this appears an unattainable ideal but I would welcome your views.

Yours faithfully,

Dr. A. Peter Smith.

Welwyn, Herts.

Dear Mrs. Teo,

In case no-one else has bothered to write in, I think I should let you know that the new program AppleWorks will not print on an Epson printer with the standard Epson 1832 interface card.

The program is designed to print on a variety of printers. Ten of the most widely used are incorporated in the program and can be called up at will. Provision is made for an eleventh, the user's own custom printer, which he will have to configure himself. The Epson MX series is included as standard but, when called upon to print, the program behaves as though printing is taking place but nothing comes out on paper.

My dealer was surprised to hear of this, as it works on his own Epson. He then realised that he was using a Grappler card and when he reverted to the 1832 card he found the same problem. Facing Apple UK with the situation, they finally admitted that there is a bug which they are endeavouring to fix but cannot suggest any timescale. They were apparently very reluctant to admit to a

fault, trying to suggest that there was a problem with the hardware. Perhaps BASUG would like to have a go at Apple on my behalf.

Yours sincerely,

Stuart F. Wood.

Reinventing the Wheel

by Yvette Raikes

Like most people, I have made the mistake of programming first and worrying afterwards. It's not just a case of bad documentation but often of bad structuring and lack of thought. Insufficient planning can cost hours of correction. Not giving a few minutes of forethought can give days (and nights!) of trouble.

Why bother to tell you this? After all, everyone does the same. Actually writing the code is far more exciting than thinking about it and you have the satisfaction of seeing your efforts in writing. But I have finally realised that to save time you must spend time and if one person can learn from my mistakes then this will be worth writing.

The first thought for a program should be a definition of the problem it is to solve. It's surprising how easy it is to lose sight of the purpose of the program in the thrill of constructing a clever subroutine. Having decided on that, you should then give a rough outline of what that problem involves. For instance, making a cup of tea involves filling a kettle, boiling the water, getting a clean cup, getting the teapot clean and warmed, putting tea in the teapot, followed by boiling water and a time delay (while the tea brews), putting milk in the cup (if desired), pouring out the tea and adding sugar and/or lemon (if desired).

Having established that, it is then easier to plan the individual areas of the problem. It becomes obvious, for example, that user choice must be allowed at some points, also that some choices preclude others, i.e. if milk, no lemon. Some stages must be completed before the next can commence. Others can happen at various points. (A clean cup can be obtained at any point until the choice for putting milk in arrives.

This is where the use of a flowchart comes into its own to show how each section

relates to all the others. I always hate writing flowcharts but find that when they are in front of me, I use them extensively and usually write better programs. A flowchart also highlights any area which has many entry and exit points, i.e. is connected with several other sections of the program, which forces you to plan how they will interrelate.

The next stage is to define how each section will be constructed. Getting a clean cup seems simple but what if there is no clean cup? Should this section include giving up? Finding a used cup and washing it up? Or should there be an option to continue with the used cup? Hygiene apart, this is where most programs collapse. Error conditions and incorrect usage can often blow a program simply because the programmer did not cater for every case. It is not enough for the program to be only able to cope with the obvious and the straightforward. It must be able to come to a satisfactory conclusion even if the problem is left unsolved e.g. if a cup can't be found, then the tea won't be made. Nevertheless, the program should not crash, it should clearly state what went wrong and if this means that it has to be aborted, it should do so tidily (switch off kettle, etc.). All too often average programs could be good if only error conditions were properly coped with. If files need closing, they should be closed. Data should be saved where necessary. In fact, wherever possible, programs should leave everything in the best possible condition for recovery and further use.

Having decided how each section should come together, it is worth checking whether there is duplicate logic in different sections which could more conveniently be written in a subroutine. Error conditions are a prime candidate for this. Disk errors and validation checks can occur in several different areas. You can often save hours of duplication by a few minutes spent here. It is irritating to discover that after inventing the wheel, you have then re-invented it elsewhere.

At this point it is usually possible to start coding, but unless it is really simple stuff, it is often wise to write your program away from the machine. It is easier to check back through your logic if you can lay all your sheets of paper out in front of you. It is more difficult when only part of the routine is visible on the screen. It can also save you hours of retyping in bits of code. At the risk of being boring, I cannot emphasise too much the need for clear and

plentiful remarks. It is amazing how quickly you can forget what a piece of coding was for. It makes debugging simpler and means that anyone else looking at your program has a fighting chance of understanding what is going on. I've often been asked to help eliminate a bug and taken a lot longer to find it than was really necessary simply because the code was not self-explanatory and there were no comments. This is also a good case for using meaningful labels.

Having (eventually) typed the program into the machine section by section, it is always worth printing a hard copy if you have the facility. It makes debugging a lot easier. You should also save your program each time a new section is entered so that if anything happens, you don't have to retype the entire thing. Where possible try testing each section separately. If you try the whole lot and it doesn't work, there's an awful lot to check through whereas if each section is debugged and working you know you have problems in the connections between them. You should save your program each time you amend it and of course, keep a backup copy. If you make a great many alterations, a new hard copy is advisable.

Finally testing the program should be an exhaustive procedure. Every choice should be tried. Invalid data should be entered. You should try to make the program fail by every method at your disposal, including putting all unlikely combinations together. If you can't crash your program, give it to someone else to try. If the program still comes out tops, then you've probably got it right. This is the time to make a final hard copy. Best of all, when the program needs updating or you want to make a change, it will be simple and straightforward because you will have all the documentation (you won't have thrown it away, will you?) and your program will be easily altered.

Advertisers

Apple Games Exchange	33
Berkshire Micros	24
Dering Systems	17,29
Elite Software Company	11,21,35
Greengate Productions	39
Lux Computer Services Ltd.	2
James Nalty	9
P & P Micro Distributors Ltd.	48
Systemics Ltd.	2
Small Ads.	9

DIARY

October

- 2nd Herts Group. 8pm. Macintosh demonstration.
A look at some of the software.
- 3rd Essex Group. 8pm.
- 8th Hants & Berks Group. 7.30pm. Databases.
What are they? Some examples.
- 12th Mid-Apple (Birmingham). 8pm.
- 15th Croydon Group - Music on Apples, e.g. Zapple. 7pm.
- 17th Essex Group. 8pm.
- 20th Assembly Language Course, County Hall, London. 9am to 5pm.
£20 members, £25 non-members.

November

- 6th Herts Group. 8pm. Which DOS? A look at ProDOS, Fast DOS,
Diversi-DOS, etc. Why not bring your favourite DOS?
- 7th Essex Group. 8pm.
- 9th Mid-Apple (Birmingham). 8pm.
- 12th Hants & Berks Group. 7.30pm. Home servicing and repairs.
Your faults and preventive maintenance.
- 19th Croydon Group - Word Processing packages compared. 7pm.
- 21st Essex Group. 8pm.
- 22nd - 24th Northern Computer Fair, Manchester.

December

- 4th Herts Group. 8pm. Communications - A look at ASCII Express
and Prestel editing in colour on the Apple.
- 5th Essex Group. 8pm.
- 10th Hants & Berks Group. 7.30pm.
- 14th Mid-Apple (Birmingham). 8pm.
- 17th Croydon Group - Adventure Games: Philosophy of design. 7pm.
- 19th Essex Group. 8pm.

If you would like your events in the diary, please write in and tell us about them.

Advertising Rates

Full page	£50.00
Half page	£27.50
1/4 page	£17.50
Flysheets	£75.00

Preparation of artwork from £ 5.00

Please send complete camera-ready artwork in monochrome. If the original is in A4, then the typeface must stand photographic reduction to A5. We can undertake minor alterations to copy.

Copy Dates

Date	Edition
October 26th	December
January 4th	February
March 1st	April
May 3rd	June
July 5th	August

TRADE IN - TRADE UP!

TO AN APPLE IIe, APPLE IIc OR A MACINTOSH.



Apple's Macintosh

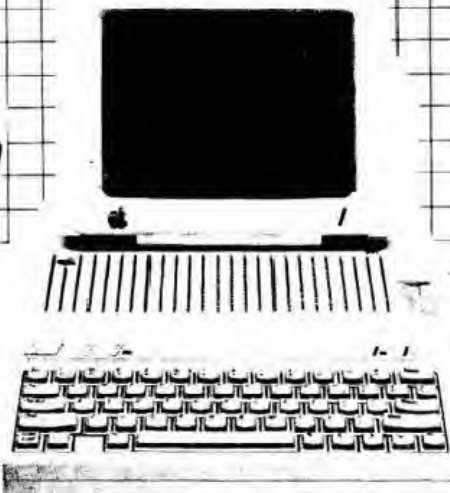
WE WILL TAKE YOUR TRUSTY AIDE
IN PART EXCHANGE FOR A NEW APPLE IIe, APPLE IIc,
OR A MACINTOSH. THE TRADE WE CAN GIVE WILL BE
DEPENDENT ON THE CONFIGURATION AND AGE OF YOUR
PRESENT MACHINE.
INTERESTED? CALL STEVE McLEAN TODAY ON
ROSSENDALE (0706) 217744 FOR IMMEDIATE
ATTENTION.



The Apple IIc showing the Side-Mounted
Half-Height Disk Drive.

OK, SO YOU LIKE YOUR PRESENT APPLE A LOT.
BUT, THERE CAN BE MANY ADVANTAGES IN MOVING
UP TO ONE OF APPLE'S NEW, POWERFUL AND PRACTICAL
MACHINES - IF ONLY THE PROBLEM OF WHAT TO DO
WITH YOUR PRESENT MACHINE CAN BE SOLVED.
WE HAVE THE ANSWER.

WE NEED SUPPLIES OF APPLE II+ AND IIe WE WILL LOVINGLY
CLEAN, POLISH AND RESTORE THEM TO LIKE NEW, BEFORE
FINDING THEM NEW HOMES. SO WE WILL MAKE ALL YOU
APPLE OWNERS AN OFFER.



The Apple IIc with IIc Monitor & Stand.

NEW APPLE PRICES

- Apple IIc Computer-£925 External Disk Apple IIc-£230
- Monitor Apple IIc-£140 Monitor Stand Apple IIc-£27
- Mouse Apple IIc (inc. Mouse Paint)-£70
- Carrying Case Apple IIc-£27 Additional Power Supply Apple IIc-£27
- Imagewriter Acc. Kit Apple IIc-£38
- Appleworks-£175 Logo Apple II 128K-£75
- Access Apple II-£57
- Apple IIe 64K Computer-£587
- Apple IIe 64K Computer (Disk Drive with Controller)-£795
- Apple IIe 64K Computer (Duodisk with Acc. Kit IIe, Monitor IIe)-£1095
- Macintosh Computer with Tree Macwrite/Macpaint-£1795
- Macintosh External Drive-£349 Macintosh Numeric Keypad-£69
- Macintosh Security Kit-£34 Macintosh Carry Case Deluxe-£69
- Macintosh Carry Case Standard-£39
- Imagewriter Printer 10"-£385
- Imagewriter Printer 15"-£525
- I/Writer Acc. Kit Macintosh-£38

Head Office:
NEW HALL HEY RD., ROSSENDALE, LANCs., BB4 6JG Tel: (0706) 217744 Telex: 635740 PETPAM G
1 GLENEAGLE RD., LONDON, SW16 6AY Tel: 01-677 7631/01-769 1022 Telex: 923070 PPCOMP G
DALE ST., BILSTON, WEST MIDLANDS, WV14 7JY Tel: 0902-43913
Norwegian Agent: Programvare Huset, Økernveien 145, N-0505 5, Tel: 47 2 64 55 77

Lines open for orders
8 a.m. - 6 p.m. Mon - Fri (Lancs)
9 a.m. - 6 p.m. Mon - Fri (London)
9.30 a.m. - 3 p.m. Sat (both offices)

